

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

In re Patent Application of:

Takahisa HATAKEYAMA et al.

Application No.: (Unassigned)

Group Art Unit:

Filed: July 9, 2003

Examiner:

For: OPEN GENERIC TAMPER RESISTANT CPU AND APPLICATION SYSTEM THEREOF

**SUBMISSION OF CERTIFIED COPY OF PRIOR FOREIGN  
APPLICATION IN ACCORDANCE  
WITH THE REQUIREMENTS OF 37 C.F.R. § 1.55**

Commissioner for Patents  
PO Box 1450  
Alexandria, VA 22313-1450

Sir:

In accordance with the provisions of 37 C.F.R. § 1.55, the applicant(s) submit(s) herewith a certified copy of the following foreign application:

Japanese Patent Application No(s). PCT/JP02/06955

Filed: July 9, 2002

It is respectfully requested that the applicant(s) be given the benefit of the foreign filing date(s) as evidenced by the certified papers attached hereto, in accordance with the requirements of 35 U.S.C. § 119.

Respectfully submitted,

STAAS & HALSEY LLP

Date: 7/9/03

By: 

J. Randall Beckers  
Registration No. 30,358

1201 New York Ave, N.W., Suite 700  
Washington, D.C. 20005  
Telephone: (202) 434-1500  
Facsimile: (202) 434-1501

JAPAN PATENT OFFICE

This is to certify that the annexed is a true copy of the following application as filed with this office.

Date of Application: July 9, 2002

Application Number: PCT/JP02/06955

Applicant(s): FUJITSU LIMITED

April 8, 2003

Commissioner,

Japan Patent Office Shinichiro Ota

Certificate No. H14-500078

# 日 本 国 特 許 庁

JAPAN PATENT OFFICE

別紙添付の書類は下記の出願書類の謄本に相違ないことを証明する。  
This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日  
Date of Application: 2002年 7月 9日

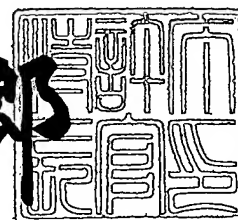
出 願 番 号  
Application Number: PCT/J P 0 2 / 0 6 9 5 5

出 願 人  
Applicant (s): 富士通株式会社

2003 年 4 月 8 日

特許庁長官  
Commissioner,  
Japan Patent Office

太田 信一郎



出証平 14-500078

0	受理官庁記入欄	PCT/JP02/06955
0-1	国際出願番号.	
0-2	国際出願日	09.07.02
0-3	(受付印)	PCT International Application 日本国特許庁
0-4	様式-PCT/RO/101 この特許協力条約に基づく国際 出願願書は、 右記によって作成された。	PCT-EASY Version 2.92 (updated 01.06.2002)
0-5	申立て 出願人は、この国際出願が特許 協力条約に従って処理されるこ とを請求する。	
0-6	出願人によって指定された受理 官庁	日本国特許庁 (RO/JP)
0-7	出願人又は代理人の書類記号	0251475/2555
I	発明の名称	開放型汎用耐攻撃CPU及びその応用システム
II	出願人	
II-1	この欄に記載した者は、	出願人である (applicant only)
II-2	右の指定国についての出願人である。	すべての指定国 (all designated States)
II-4ja	名称	富士通株式会社
II-4en	Name	FUJITSU LIMITED
II-5ja	あて名:	211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番1号
II-5en	Address:	1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan
II-6	国籍(国名)	日本国 JP
II-7	住所(国名)	日本国 JP
II-8	電話番号	044-754-3798
II-9	ファクシミリ番号	044-754-3536
III-1	その他の出願人又は発明者	
III-1-1	この欄に記載した者は	発明者である (inventor only)
III-1-4ja	氏名(姓名)	畠山 卓久
III-1-4en	Name (LAST, First)	HATAKEYAMA, Takahisa
III-1-5ja	あて名:	211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番1号 富士通株式会社内
III-1-5en	Address:	c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan



## 特許協力条約に基づく国際出願願書

原本(出願用) - 印刷日時 2002年07月09日 (09. 07. 2002) 火曜日 14時42分35秒

III-2 III-2-1 III-2-4j a III-2-4e n III-2-5j a  III-2-5e n	その他の出願人又は発明者 この欄に記載した者は 氏名(姓名) Name (LAST, First) あて名:  Address:	発明者である (inventor only) 丸山 秀史 MURUYAMA, Hidefumi 211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番1号 富士通株式会社内 c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan
III-3 III-3-1 III-3-4j a III-3-4e n III-3-5j a  III-3-5e n	その他の出願人又は発明者 この欄に記載した者は 氏名(姓名) Name (LAST, First) あて名:  Address:	発明者である (inventor only) 牛若 恵一 USHIWAKA, Keiichi 211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番1号 富士通株式会社内 c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan
III-4 III-4-1 III-4-4j a III-4-4e n III-4-5j a  III-4-5e n	その他の出願人又は発明者 この欄に記載した者は 氏名(姓名) Name (LAST, First) あて名:  Address:	発明者である (inventor only) 長谷部 高行 HASEBE, Takayuki 211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番1号 富士通株式会社内 c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan
III-5 III-5-1 III-5-4j a III-5-4e n III-5-5j a  III-5-5e n	その他の出願人又は発明者 この欄に記載した者は 氏名(姓名) Name (LAST, First) あて名:  Address:	発明者である (inventor only) 前田 直昭 MAEDA, Naoaki 211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番1号 富士通株式会社内 c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan

III-6 III-6-1 III-6-4j a III-6-4e n III-6-5j a	その他の出願人又は発明者 この欄に記載した者は 氏名(姓名) Name (LAST, First) あて名:	発明者である (inventor only) 須賀 敦浩 SUGA, Atsuhiro 211-8588 日本国 神奈川県 川崎市 中原区上小田中4丁目1番1号 富士通株式会社内 c/o FUJITSU LIMITED, 1-1, Kamikodanaka 4-chome, Nakahara-ku, Kawasaki-shi, Kanagawa 211-8588 Japan
III-6-5e n	Address:	
IV-1  IV-1-1ja IV-1-1en IV-1-2ja  IV-1-2en  IV-1-3 IV-1-4 IV-1-5	代理人又は共通の代表者、通知 のあて名 下記の者は国際機関において右 記のごとく出願人のために行動 する。 氏名(姓名) Name (LAST, First) あて名:  Address:  電話番号 ファクシミリ番号 電子メール	代理人 (agent)  大菅 義之 OSUGA, Yoshiyuki 102-0084 日本国 東京都 千代田区 二番町8番地20 二番町ビル3F 3rd Fl., Nibancho Bldg., 8-20, Nibancho, Chiyoda-ku, Tokyo 102-0084 Japan 03-3238-0031 03-3238-0034 osugapat@osuga-pat.com
V V-1	国の指定  広域特許 (他の種類の保護又は取扱いを 求める場合には括弧内に記載す る。)	AP: GH GM KE LS MW MZ SD SL SZ TZ UG ZM ZW 及びハラレプロトコルと特許協力条約の締約国である 他の国 EA: AM AZ BY KG KZ MD RU TJ TM 及びユーラシア特許条約と特許協力条約の締約国であ る他の国 EP: AT BE BG CH&LI CY CZ DE DK EE ES FI FR GB GR IE IT LU MC NL PT SE SK TR 及びヨーロッパ特許条約と特許協力条約の締約国であ る他の国 OA: BF BJ CF CG CI CM GA GN GQ GW ML MR NE SN TD TG 及びアフリカ知的所有権機構と特許協力条約の締約国 である他の国
V-2	国内特許 (他の種類の保護又は取扱いを 求める場合には括弧内に記載す る。)	AE AG AL AM AT AU AZ BA BB BG BR BY BZ CA CH&LI CN CO CR CU CZ DE DK DM DZ EC EE ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NO NZ OM PH PL PT RO RU SD SE SG SI SK SL TJ TM TN TR TT TZ UA UG UZ VN YU ZA ZM ZW

## 特許協力条約に基づく国際出願願書

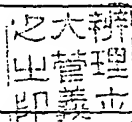
0251475/2555

原本（出願用）～印刷日時 2002年07月09日（09.07.2002）火曜日 14時42分35秒

V-5	指定の確認の宣言 出願人は、上記の指定に加えて、規則4.9(b)の規定に基づき、特許協力条約のもとで認められる他の全ての国の指定を行う。ただし、V-6欄に示した国の指定を除く。出願人は、これらの追加される指定が確認を条件としていること、並びに優先日から15月が経過する前にその確認がなされない指定は、この期間の経過時に、出願人によって取り下げられたものとみなされることを宣言する。		
V-6	指定の確認から除かれる国	なし (NONE)	
VI	優先権主張	なし (NONE)	
VII-1	特定された国際調査機関 (ISA)	日本国特許庁 (ISA/JP)	
VIII	申立て	申立て数	
VIII-1	発明者の特定に関する申立て	-	
VIII-2	出願し及び特許を与えられる国際出願日における出願人の資格に関する申立て	-	
VIII-3	先の出願の優先権を主張する国際出願日における出願人の資格に関する申立て	-	
VIII-4	発明者である旨の申立て（米国を指定国とする場合）	-	
VIII-5	不利にならない開示又は新規性喪失の例外に関する申立て	-	
IX	照合欄	用紙の枚数	添付された電子データ
IX-1	願書（申立てを含む）	5	-
IX-2	明細書	97	-
IX-3	請求の範囲	11	-
IX-4	要約	1	EZABST00.TXT
IX-5	図面	52	-
IX-7	合計	166	
	添付書類	添付	添付された電子データ
IX-8	手数料計算用紙	✓	-
IX-11	包括委任状の写し	✓	-
IX-17	PCT-EASYディスク	-	フロッピーディスク
IX-18	その他	納付する手数料に相当する特許印紙を貼付した書面	-
IX-18	その他	国際事務局の口座への振込みを証明する書面	-
IX-19	要約書とともに提示する図の番号	3	
IX-20	国際出願の使用言語名:	日本語	
X-1	提出者の記名押印		
X-1-1	氏名(姓名)	大菅 義之	

受理官庁記入欄

10-1	国際出願として提出された書類の実際の受理の日	09.07.02
------	------------------------	----------



## 特許協力条約に基づく国際出願願書

原本（出願用） - 印刷日時 2002年07月09日（09. 07. 2002）火曜日 14時42分35秒

10-2	図面：	
10-2-1	受理された	
10-2-2	不足図面がある	
10-3	国際出願として提出された書類を補完する書類又は図面であつてその後期間内に提出されたものの実際の受理の日（訂正日）	
10-4	特許協力条約第11条(2)に基づく必要な補完の期間内の受理の日	
10-5	出願人により特定された国際調査機関	ISA/JP
10-6	調査手数料未払いにつき、国際調査機関に調査用写しを送付していない	

## 国際事務局記入欄

11-1	記録原本の受理の日	
------	-----------	--

## 明細書

## 開放型汎用耐攻撃CPU及びその応用システム

## 5 技術分野

本発明は、機器内部の情報を改ざんしたり、秘密情報を取り出したりしようとする攻撃に対して対抗する技術に関する。

## 背景技術

10 近年、デジタルコンテンツの流通の促進に伴い、著作権などのコンテンツに係わる権利を保護する技術（DRM：（Digital Rights Management）が提供されている。DRMの例として、例えば、三洋電機、日立、富士通の3社が共同開発したUDAC（Universal Distribution with Access Control）が挙げられる。

15 UDACのようなDRMを実装する際、コンテンツ保護のセキュリティを十分なレベルにするためには、利用者のシステムにおいてDRMをTRM（Tamper Resistant Module）とすることが重要である。（以下、DRMをTRMとすることをDRMのTRM化という。） 現在、コンシューマ向けの製品で一般に実施されるTRM化には、大きく分けてハードウェアTRM及びソフトウェアTRMの2つの方法がある。

ハードウェアTRMとは、半導体の外部端子から秘密情報の読み出しなどができない構造にし、特殊コーティングや極微細化などを施すことによって実現されたTRMである。ハードウェアTRMをソフトウェアTRMと比較して、特に「攻撃対抗容器」と呼ぶこともできる。ソフトウェアTRMとは、暗号化  
25 させたいコードを解析困難な構造にして暗号化し、そのコードを実行の瞬間に

復号することによって実現されたTRMである。

しかし、従来のTRMには、以下のような問題があった。

### 1. ハードウェアTRMの問題

- ・コンシューマ向け用途ではコストを重視する関係上DRMを半導体パッケージ化する必要があるが、一つのチップに盛り込むことができるリソースには限りがあるため、多様な機能を提供することはできない。ライセンスキーやCRL (Certificate Revocation List) を記録する領域の大きさを拡張したい場合や、コンテンツ利用条件をXrML (eXtensible rights Markup Language) で表現したい場合などには、ハードウェアTRMでは対応できない。
- 10   ・安全性を重視すると、利用するプロセッサと記録媒体をすべてハードウェアTRM内に持つ必要があるが、このため製造後に機能を拡張させることが困難になっている。また、このことからソフトウェアが利用できる記録媒体が限定されるため、リソース消費が高い一般の（保護の必要がない）ソフトウェアを同時にプロセッサに実行させることは困難である。また、暗号鍵が異なる複
- 15   数の保護ソフトウェアを同時にプロセッサに実行させることも困難である。つまり、ハードウェアTRMには汎用性がない。

・コンテンツの種類の追加やDRMのバージョンアップごとに新たなチップの開発し、新たに半導体製造工程などのラインを起こすことが必要となるため製造コストが大きくなる。

- 20   ・互換性を持つ汎用CPUチップ上にそれぞれのソフトウェアを開発・搭載するだけで何種類もの機能を利用者に提供することによりスムーズなバグ修正や機能拡張も実現してきた「情報技術の逐次進化」を阻害しかねなかった。

### 2. ソフトウェアTRMの問題

- ・おもとの秘密鍵は暗号化することができず、どのように分散などして保
- 25   管してもソフトウェア解析だけで容易に秘密鍵を見つけ出すことができる。

・ハードウェア I C E (In Circuit Emulator) を用いれば実行内容をトレースすることにより、秘密鍵を始めとする各種秘密情報を容易に見つけ出すことができる。

## 5 発明の開示

本発明の目的は、上記問題を解決し、ソフトウェア T R M 並みの汎用性を持ちつつも、ハードウェア T R M 並みの安全性を有する T R M を提供することである。

上記目的を達成するために、本発明の 1 態様によれば、中央演算装置において、秘密裏に隠された第 1 の秘密鍵と、暗号化及び復号を行う暗号手段とを備え、前記第 1 の秘密鍵は、公開鍵と対であり、前記暗号手段は、前記公開鍵を用いて暗号化された第 1 のプログラムの第 1 のライセンスを前記第 1 の秘密鍵を用いて復号することにより、前記第 1 のライセンスから前記第 1 のプログラムを復号するためのコード復号鍵を取得するように構成する。なお、コード復号鍵は、第 1 のプログラムを暗号化する際に使用されたコード暗号鍵と同じであつてもよい。また、第 1 のライセンスは、第 1 のプログラムに埋め込まれる事としてもよいし、両者は別々に流通する事としても良い。

このように構成することにより、前記第 1 のプログラムを復号するための鍵を得るために必要な秘密鍵は、中央演算装置内に秘密裏に記録されているため、分散されて保管されることはない。従って、プログラムを解析する事によって秘密鍵が容易に発見されてしまうという問題を解決する事ができる。

また、上記中央演算装置は、キャッシュを更に備え、前記暗号手段は、前記第 1 のプログラムを構成する暗号化ブロックがメモリ領域から前記キャッシュに出力される際に、キャッシュ単位で前記暗号化ブロックを復号する、こととしてもよい。キャッシュ単位で暗号化ブロックを復号し、キャッシュに記録す

ることにより、従来より安全性を向上させることが可能となる。

- さらに、また、上記中央演算装置は、ユーザが参照したり改ざんしたりすることができない耐攻撃バッファを更に備え、前記コード復号鍵は、前記耐攻撃バッファに記録されることとしてもよい。耐攻撃バッファ内に記録された鍵は、
- 5    カーネルモードであってもユーザが参照したり改ざんしたりする事ができないため、コード復号鍵を安全に保持することが可能となる。

- ここで、前記耐攻撃バッファは、さらに、外部出力禁止情報及びキャッシュロック情報を含み、前記外部出力禁止情報は、対応する前記耐攻撃バッファ内の情報を前記耐攻撃バッファ外に出力しても良いか否かを示し、前記キャッシュ
- 10    ュロック情報は、対応する情報をキャッシュ外に出力しても良いか否かを示し、
- 前記外部出力禁止情報及び前記キャッシュロック情報に基づいて、第1のプログラム及び他のプログラムとの間における前記第1のライセンスの移動は管理されることとしてもよい。これにより、前記中央演算装置に、複製不能プライベートロッカーを実現させる事が可能となる。従って、例えば、複数のプロ
- 15    グラム間においてライセンスを移動可能とする場合であっても、再送攻撃によるライセンスの不正コピーを防止することが可能となる。

- また、上記中央演算装置において、前記第1のライセンスは、前記第1のプログラムの実行プロセスがメモリ領域にアクセスする際のアクセス条件を含み、前記第1のプログラムを構成する暗号化ブロックが記録される前記メモリ領域
- 20    のアドレスと前記メモリ領域へのアクセス条件とを記録するTLB (Translation Lookaside Buffer) と、メモリ管理手段と、キャッシュと、プロセッサコアを更に備えるように構成してもよい。この構成において、前記耐攻撃バッファは前記TLBとリンクされ、前記メモリ管理手段は、前記暗号化ブロックを記録するメモリ領域のアドレスに基づいて前記TLBから前記メモリ
- 25    領域へのアクセス条件を取得し、さらに、前記耐攻撃バッファから、前記メ



メモリ領域に対応する前記コード復号鍵を取得する。前記プロセッサコアは、前記メモリ管理手段が取得した前記アクセス条件に基づいて、前記実行プロセスから前記メモリ領域へのアクセスを実行することが許可されているか否か判定し、前記メモリ領域へのアクセスを実行する事が許可されていると判定した場合、前記実行プロセスから前記メモリ領域へのアクセスを実行する。前記暗号手段は、前記メモリ管理手段が取得した前記コード復号鍵を用いて前記メモリ領域内の前記暗号化ブロックを復号して得られたコードを前記キャッシュに書き込む。

メモリ領域は、TLBを介して耐攻撃バッファと対応付けられており、そのメモリ領域に書きこまれたブロックを復号するための鍵は、この対応関係に基づいて取得される。さらに、メモリ領域へのアクセスも、TLB内のアクセス条件に従って制御される。従って、安全にメモリ領域へのアクセス制御及びブロックのキャッシュへのロードを行うことが可能となる。

ここで、前記第1のプログラムの実行プロセスからアクセスされるメモリ領域が、第1のメモリ領域から第2のメモリ領域に切り替わった場合、前記メモリ管理手段は、さらに、前記耐攻撃バッファから取得された前記第1のメモリ領域に対応するコード復号鍵と、前記第2のメモリ領域に対応するコード復号鍵とが一致するか否か判定し、一致すると判定した場合、前記実行プロセスから前記第2のメモリ領域へのアクセスを実行し、一致しないと判定した場合、前記実行プロセスから前記第2のメモリ領域へのアクセスを実行しない、こととしてもよい。これにより、Call、Jump または Return 等の命令によって、あるメモリ領域から、そのメモリ領域に対応するコード復号鍵と異なる他のコード復号鍵が対応している他のメモリ領域に実行中アドレスが移動した際には、その実行プロセスから上記他のメモリ領域にはアクセスできないこととなる。これにより、あるオーナープロセスが、他のオーナープロセスのメモリ領域へ

アクセスすることを禁止することが可能となる。

さらにまた、上記中央演算装置において、前記耐攻撃バッファには前記コード暗号鍵ごとに、異なるデータ暗号鍵が記録され、前記暗号手段は、前記データ暗号鍵を用いて前記キャッシュ内のデータを暗号化した後、前記TLBを介して前記データ暗号鍵に対応付けられたメモリ領域に記録し、前記メモリ領域から暗号化されたデータを読み出して前記データ暗号鍵を用いて復号した後、前記キャッシュに書き込むこととしてもよい。これにより、データをメモリ領域に退避させたり、キャッシュにロードさせたりする操作を、安全に行う事が可能となる。

10      また、上記中央演算装置において、第1のコードを実行することにより得られた作業データを第2のコードで利用する場合、前記プロセッサコアは、前記作業データを記録するメモリ領域へのアクセス権を前記第2のコードに与えるように前記TLBを設定し、且つ、前記作業データを暗号化するためのコード暗号鍵を、前記第2のコードが前記作業データを前記メモリ領域から読み出す  
15      際に使用するよう前記耐攻撃バッファを設定することとしてもよい。

これにより、通常、第1のコードのコード暗号鍵と、第2のコードのコード暗号鍵が異なる場合であっても、第1のコードと第2のコードとがデータ暗号鍵を共用することにより第1のコードを実行した結果得たデータを第2のコードが読み出す事が可能となる。従って、前記中央演算装置が2以上のソフトウェアを保護しながら実行する場合であっても、必要に応じて安全にメモリ領域を共有する事が可能となる。

また、上記中央演算装置において、レジスタと、レジスタへのアクセス制御を行うためのレジスタアクセス制御テーブルと、を更に備え、前記プロセッサコアは、前記レジスタアクセス制御テーブル内の封印フラグを用いて、前記レジスタの封印及び解放を制御する、こととしてもよい。これにより、プロセス  
25

図34は、DRMソフトウェアモジュールの構築モデルの一例を示す図である。

図35は、メディアDRMによって行われる処理を示すフローチャート（その1）である。

5 図36は、メディアDRMによって行われる処理を示すフローチャート（その2）である。

図37は、デコーダDRMによって行われる処理を示すフローチャート（その1）である。

10 図38は、デコーダDRMによって行われる処理を示すフローチャート（その2）である。

図39は、デコーダDRMによって行われる処理を示すフローチャート（その3）である。

図40は、再生アプリケーションによって行われる処理を示すフローチャートである。

15 図41は、秘密情報を維持・管理する方式を示す図である。

図42は、ライセンス管理のためのメモリアクセスの方式を示す図である。

図43は、第2実施形態に係わるGTを実現するCPUの構成図である。

図44は、第2実施形態におけるブロックの構造を示す図である。

図45は、ebimが4である場合のブロックの構造を示す図である。

20 図46は、保護ブロックセクタによって行われる処理を示すフローチャートである。

図47は、ハッシュエンジンによって行われる処理を示すフローチャートである。

25 図48は、ハッシュチェッカによって行われる処理を示すフローチャートである。

図 4 9 は、NOP の処理について説明する図である。

図 5 0 は、マルチ CPU の構成図である。

図 5 1 は、コードオブジェクトから保護コード実行形式を出力するツール群の例を示す図である。

- 5 図 5 2 は、GT を、パーソナルコンピュータ又はワークステーションに実装した場合のシステム構成例を示す図である。

#### 発明を実施するための最良の形態

- 10 本発明の実施形態について図面を参照しながら説明する。なお、同じ装置等には同じ参照番号をつけ、説明を省略する。

本発明は、CPU に汎用 TRM 機能を実装する事により、CPU を、汎用性を有する攻撃対抗容器とする。以下の説明において、本発明に係わる汎用 TRM 機能を実装する CPU を「Generic TRM」といい、GT と略称する。

#### <構成>

- 15 GT の構成について説明する前に、GT と、その GT によって実行されるソフトウェアの利用関係について説明する。図 1 に、GT を実現する CPU パッケージとその CPU パッケージによって実行される GT 基本パッケージソフトウェアの利用関係モデルを示す。

- 20 図 1 に示すように、GT 基本パッケージソフトウェアは、アプリケーション層、DRM 層、PKI ライブラリ層に分けることができる。アプリケーション層には、保護されるアプリケーション（以下、保護アプリケーションという）が含まれる。DRM 層には、デコーダ DRM 3 0 及びメディア DRM 4 0 が含まれる。PKI ライブラリ層には、PKI ライブラリ 2 0 が含まれる。デコーダ DRM 3 0、メディア DRM 4 0 及び PKI ライブラリ 2 0 は、OS  
25 (Operation System) の一部である。

各基本ソフトウェアパッケージはそれぞれ次のような機能を持つ。なお、これらの機能は、従来から知られているため説明は省略する。

PKIライブラリ：

- ・標準のPKIX (Public Key Infrastructure (X.509))ライブラリ

5   メディアDRM (Virtual Media DRM, Trusted Server DRMを含む)：

- ・ライセンス生成・管理・削除・移動・複製の制御
- ・保護コンテンツ管理
- ・UDAC-MB/LB/PI 認証・ライセンス管理機能
- ・ライセンス変換機能：MB (Media Base)、PI (Protocol Independent)、

10   GT間での変換など

デコーダDRM：

- ・メディアDRMからのライセンス取得
  - ・許諾ライセンス復号とコンテンツ復号鍵の維持
  - ・コンテンツの復号、汎用利用制御とアプリケーションへの安全な転送
- 15   ・利用制御機能拡張機能

図1に示すように、各DRM30及び40は、ソフトウェアとして実現され、ハードウェアとして実現されるのではない。また、GT10はCPUとして実現される。GT10 (CPU) は、DRM30及び40、PKIライブラリ20及び保護アプリケーション50を実行する。

20   図2に、図1に示す基本パッケージ以外のモジュールを含めたGT上でのプログラミングモデルを示す。

図2に示すように、OSには、上述のPKIライブラリ20、デコーダDRM30及びメディアDRM40に加えて、OSカーネル60及びデバイスドライバ70を備える。デバイスドライバ70は、周辺ハードウェア60を動作さ

25   せるために必要なソフトウェアである。

保護アプリケーション50及び保護されないアプリケーションを含む複数のアプリケーションは、このOS上で動作する。つまり、GT10上では、DRM機能を用いた保護アプリケーションとともに、従来のCPU上で稼動する他のアプリケーションも稼動する。このように、GT10は、一般の保護の必要  
5 がないアプリケーションを保護アプリケーションと同時に実行することが可能な汎用性を有する耐攻撃容器である。

なお、OSカーネル60は、従来の動作に加え、GT10において、メモリ管理及びコンテキストの切り替え制御の際に、レジスタ封印等のGT10に固有の処理（後述）を行う。しかし、それらの処理をOSカーネル60に行わせることは、GT10のセキュリティに影響を及ぼさない。すなわち、OSカー  
10 ネル60にセキュリティホールが存在している場合であっても、GT10による保護の安全性には影響はない。

デコーダDRM30、メディアDRM40及びこれらのDRMが用いるPKIライブラリ20内の暗号・復号ライブラリ、更には、TRM化が必要なアプリケーションは、GT10上で保護されるGT保護コードとして流通し、実行  
15 される必要がある。そして、これらのソフトウェアモジュールは、ほぼ全文を暗文にされる必要がある。

上述のように、従来、ソフトウェアTRMには、拡張性はあるが容易に破壊できるという性質があった。しかし、本発明によれば、GT10によって保護  
20 されるDRMソフトウェアモジュールを採用する事により、DRMソフトウェアモジュールにハードウェアTRM並みの強度を与える事を可能としている。一方で、ハードウェアTRMには拡張性が無く、リソースも限られるという問題があったが、GT10によれば、DRMソフトウェアを採用しているため、機能的な拡張には、GT10を搭載するCPUに変更を加える必要は無く、D  
25 RMソフトウェアのバージョンアップによって対応する事が可能である。

DRM構築モデルは、UDAC-MB (Universal Distribution with Access Control-Media Base)UDAC-LA、UDAC-PIそれぞれのアーキテクチャ及び仕様に従うこととしてもよい。

以下、図3を用いて、第1実施形態に係わるGTを実現するCPUの構成について、データのやり取りについて言及しながら説明する。

図3に、平文又は暗文のみからなるコードブロック或いはデータブロックを想定した場合の (ebim=0, 1 のみをサポートする場合)、GT内部の回路ブロックの構成と回路ブロック間のデータ交換モデルを示す。

GT10は、プロセッサコア11、命令キャッシュ12、データキャッシュ13、命令MMU (Memory Management Unit) 14、データMMU15、暗号エンジン16を備え、RAM (Random Access Memory) 17に接続されている。

本発明に係わるGT10の特徴の1つとして、コードブロック又はデータブロックを暗号化したり、復号したりする暗号エンジン16を備え、この暗号エンジン16を用いて暗号化されたコードブロックやデータブロックを復号してからGT10内部のキャッシュ12又は13に保持し、データキャッシュ12からRAM17に作業データを出力する際には、暗号エンジン16を用いてそのデータを暗号化してから出力することがある。以下、GT10におけるブロック間のデータ交換について説明する。

まず、RAM17から入力されたコードブロックは、プロセッサコア11及び命令MMU14によって保護すべきコードブロック (以下、保護コードブロックという) 又は平文コードブロックのいずれであるか識別される。RAM17から入力されたデータブロックは、プロセッサコア11及びデータMMU15によって保護すべきデータブロック (以下、保護データブロックという) 又は平文データブロックのいずれであるか識別される。

保護コードブロック及び保護データブロックは暗号エンジン16に送信され

るよう、それぞれ命令MMU 14及びデータMMU 15からRAM 17に対して保護ブロック転送先のアドレスの指定がされる。暗号エンジン16に出力されたコードブロックやデータブロックは復号されて、それぞれ命令キャッシュ12やデータキャッシュ13に出力される。保護コードブロックを復号する際  
5 に用いられる鍵 Kc 及び保護データブロックを復号する際に用いられる鍵 Kd は、プロセッサコア11から暗号エンジン16に出力される。平文コードブロック又は平文データブロックは、そのまま、それぞれ命令キャッシュ14又はデータキャッシュ15に出力される。

また、保護コードブロック及び保護データブロックの安全性を確保するため  
10 に、GT 10によれば、作業データをデータキャッシュ13からRAM 17に出力する際に、データキャッシュ13の出口において作業データを暗号化する。そのために、まず、データキャッシュ13から作業データが暗号エンジン16に出力されるとともに、プロセッサコア11から作業データを暗号化するための鍵 Kd（保護データブロックを復号する際に用いる鍵と同じ）が暗号エンジン  
15 16に出力される。暗号エンジン16は、その鍵 Kd を用いてその作業データを暗号化し、暗号化された作業データをRAM 17等に設けられた仮想メモリ領域に出力する。さらに、暗号化する際に、データブロックに乱数を付加することとしても良い。

なお、データブロックを仮想メモリ領域に出力する際には、そのデータブロックが保護されるべきデータブロックであるか否か、TLB（Translation  
20 Look aside Buffer）（不図示、後述）に基づいてプロセッサコア11によって判断され、判断結果に基づいて、データキャッシュ機能からのアドレス指定の際に暗号エンジン16を通すかどうかが決定的される。また、暗号化及び暗号の復号に用いる鍵 Kc 及び Kd は、GT 10内の暗号エンジン16がGTライセンスを復号することによって得られ、耐攻撃バッファ（以下、TRB：Tamper  
25



Resistant Buffer という) (不図示、後述) に格納されている。

命令MMU 14 とデータMMU 15 は、図3において、別々のブロックとして示されているが、一つのモジュールにしてもよい。命令キャッシュ12 とデータキャッシュ13 についても同様である。

- 5    上記のように、GT10 によれば、キャッシュの出入り口で暗号化されたコードブロックやデータブロックをキャッシュ単位で復号する。ところで、CPU 内でコードを1 つずつ暗号化する場合、2 バイト程度の単位で行う必要がある。しかし、十分な強度を得るためには現在の技術では8 バイト程度を暗号化  
10    ブロックとすることが要求されるため、2 バイト程度の単位では十分な強度が維持できない。しかし、GT10 によれば、RAM17 からの暗号化されたコードブロックやデータブロックを復号する際にはキャッシュ単位で復号して維持する事により、保護コードブロック及び保護データブロックを安全に守る事が可能となる。

- 15    なお、GT10 に命令キャッシュ12 が備えられない場合、GT10 内にRAM領域を設け、全ての実行コードを内部で復号して内部のRAM領域に保存してから実行することとしてもよい。これにより、上記構成と同等の安全性を確保する事ができる。また、GT10 にデータキャッシュ13 が備えられない場合、GT10 の内部にRAM領域を設け、保護が必要な作業データをGT10 の内部のRAM領域に記録する。これにより、上記構成と同等の安全性を確保  
20    する事ができる。

- 25    また、従来、CPU 内の作業データを一時的にRAMに出力する際に、秘密にすべきデータが漏洩することがあった。また、そのデータの内容を監視することによって命令コードの推測を行うことによって、CPU 内に保持されている秘密鍵を推測することも可能であった。しかし、GT10 によれば、データ  
25    キャッシュから作業データをRAMに出力する際に暗号化し、キャッシュに復

がデータをレジスタに格納する場合には、現在実行中のプロセス以外からレジスタに格納されたデータにアクセスする事ができないように封印することが可能となる。

また、上記中央演算装置において、前記TLBの内容を外部のページテーブルに記録する際には、前記暗号手段は、記録する内容に署名を付与し、前記ページテーブルの内容を前記TLBに取り込む前には、前記暗号化手段は、前記署名が正しいことを確認する、こととしてもよい。これにより、TLBの内容を安全にページテーブルに保持させる事が可能となる。

また、上記中央演算装置において、前記耐攻撃バッファの内容を外部記憶装置内の暗号化キーテーブルに記録する際には、前記暗号手段は、記録する内容を暗号化することとしてもよい。これにより、耐攻撃バッファの内容を外部記憶装置に安全に保持する事が可能となる。

なお、TLBの内容の署名を作成する際に使用される秘密鍵、及び耐攻撃バッファの内容を暗号化する際に使用される秘密鍵は、ともに、中央演算装置内のプロセッサによって生成されることとしてもよい。

また、前記中央演算装置は、他の中央演算装置と接続され、前記中央演算装置は、前記他の中央演算装置と相互に認証を行う事にことによりセッション鍵を取得し、前記中央演算装置の前記暗号手段は、前記セッション鍵を用いて前記中央演算装置の前記キャッシュの内容を暗号化して、前記他の中央演算装置に同期転送する、こととしてもよい。これにより、上記構成を有する中央演算装置を複数有するマルチCPUにおいて、キャッシュ内の内容を安全に同期させることが可能となる。

また、上記中央演算装置において、前記暗号手段は、前記第1のプログラムを実行する前に、前記公開鍵を用いて第2のプログラムに付加された前記第2のライセンスを復号する事により第2の秘密鍵を暗号化する際に用いられた秘

密鍵暗号鍵を取得し、さらに、前記取得された秘密鍵暗号鍵を用いて前記第2の秘密鍵を復号する、こととしてもよい。

また、この際に、前記第2のライセンスには、第1のプログラムの実行プロセスから読み出しのみ可であることを示すアクセス条件が付加されており、前記第2の秘密鍵は、前記第1のプログラムの実行プロセスからの読み出しのみ可能であることとしてもよい。第2の秘密鍵を第1のプログラムの実行プロセスからのみ読むことができないこととすることにより、秘密情報を安全に維持管理することが可能となる。

なお、前記第1のプログラムは、TRM化が要求されるどのようなソフトウェアであってよい。例えば、第1のプログラムとして、トラステッドコンピューティングモジュール、前記中央演算装置に電子財布を実現させるプログラム、個人情報を扱うソフトウェア、前記中央演算装置の実装コードのウィルスチェックソフトウェア、複数の中央演算装置間を移動する移動エージェント、ロボットの制御プログラム、ICカード用のセキュリティプログラム等が挙げられる。

また、上記中央演算装置において、前記第1のプログラムを構成するブロックには様々な形態が考えられ、それに応じて中央演算装置に更なる手段を備える事としてもよい。

例えば、ブロックは、ブロックのハッシュ値の確認が必要であるか否かを示すハッシュ確認要否情報を含み、前記ハッシュ確認要否情報に基づいて、前記ブロックのハッシュ値を算出し、前記ブロックに付加するハッシュ手段と、前記ハッシュ確認要否情報に基づいて、前記ブロックの前記ハッシュ値を確認するハッシュ確認手段と、を更に備えることとしてもよい。

また、例えば、ブロックは、ブロックが保護を必要とするか否かを示す暗号化要否情報を含み、前記暗号化要否情報に基づいて、前記ブロックを前記暗号

手段に出力するか、そのままキャッシュ又はメモリ領域に出力するか判定する保護ブロック選択手段を、更に備えることとしてもよい。

また、ブロックごとにブロックを選択する保護ブロック選択手段の付加を低減するために、以下のようにしてもよい。

- 5     例えば、第1のプログラムの実行ファイルのヘッダは、前記第1のプログラムを構成するブロックの構成を示す暗号化ブロックビットマップを含み、保護ブロック選択手段は、前記暗号化ブロックビットマップに基づいて、前記ブロックを前記暗号手段に出力するか、そのままキャッシュ又はメモリ領域に出力するか判定する。

- 10    また、例えば、第1のプログラムのコードの先頭は、前記第1のプログラムを構成する複数のブロックが、平文ブロックと暗号化ブロックの組合せの繰り返しであり、前記組合せにおいて平文ブロックが連続する数及び暗号化ブロックが連続する数を指定するコードであり、

- プロセッサコアこのコードを実行することにより、前記ブロックを前記暗号  
15    手段に出力するか、そのままキャッシュ又はメモリ領域に出力するか判定する。

また、上記中央演算装置において、前記キャッシュとメモリの間に、前記暗号手段を介するキャッシュラインと、前記暗号手段を介さないキャッシュラインと、を更に備えることとしてもよい。これにより、処理の高速化を図ることができる。

- 20    また、本発明の他の態様として、中央演算装置に保護プログラムを実行する許諾を与える制御を行わせるプログラムであって、前記保護プログラムは、コード暗号鍵によって暗号化され、保護プログラムに対応して、前記コード暗号鍵を含み、且つ、前記中央演算装置に秘密裏に備えられた秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、前記中央演算装置が前記保護  
25    プログラムを実行する前に、前記ライセンスを前記中央演算装置に投入し、前

記中央演算装置に備えられた暗号手段に、前記秘密鍵を用いて前記ライセンスを復号することにより、前記ライセンスから前記コード暗号鍵を取得させ、前記暗号手段に、前記コード暗号鍵を用いて前記保護プログラムを復号させる、ことを含む処理を前記中央演算処理装置に実行させる、ように構成する。

- 5      このプログラムは、上記構成を有する中央演算装置によって行われる処理と同様の作用・効果を得ることができるものである。従って、このプログラムによっても、上記問題を解決する事ができる。また、上記プログラムを記録する記録装置も、上記プログラムを上記中央演算装置によって実行されることにより、上記構成を有する中央演算装置によって行われる処理と同様の作用・効果
- 10      を得ることができるものである。

また、上記中央演算装置を構成する各手段によって行われる動作を手順として含むソフトウェア実行許諾方法も、上記構成を有する中央演算装置によって行われる処理と同様の作用・効果を得ることができるものである。

- また、本発明の更なる1態様によれば、コンピュータにおいて実行されるプログラムであって、コード暗号鍵によって暗号化されたプログラムと、前記保護プログラムに対応して、前記コード暗号鍵を含み、且つ、前記プログラムを実行するべきコンピュータに備えられた中央演算装置が秘密裏に備える秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、前記ライセンスは、前記プログラムが実行される前に前記中央演算装置に投入され、前記中央
- 15      演算装置によって前記秘密鍵を用いて復号され、前記プログラムは、前記ライセンスから取得された前記コード暗号鍵を用いて、前記中央演算装置によって復号される、ことを特徴とする。このプログラムは、上記中央演算装置によって保護されるソフトウェアであり、上記中央演算処理装置を有するコンピュータによって実行される事によって、ハードウェアTRM並みの安全性を得ること
- 20      ができる。
- 25      とができる。

また、上記プログラムを記録する、前記コンピュータによって読み取り可能な記録媒体もまた、その記録媒体からプログラムをロードし、プログラムを上記中央演算処理装置を有するコンピュータによって実行される事によって、上記プログラムと同様の作用・効果を得ることができる。

- 5      また、本発明の更なる別の態様によれば、上記構成を有する中央演算装置を備えるコンピュータにおいて実行されるプログラムを生成するプログラム生成装置であって、コードオブジェクトを入力する入力手段と、入力された前記コードオブジェクトを複数のブロックに分割し、それぞれのブロックにNOP命令を追加するリンカ前処理手段と、アドレス解決を行うリンカ手段と、コード
- 10   暗号鍵を用いて各ブロックを暗号化することにより保護コード実行形式を生成する保護コード実行形式生成手段と、前記コード暗号鍵を含み、前記秘密鍵と対になる公開鍵によって暗号化されたライセンスを生成するライセンス生成手段とを備え、前記ライセンスは、前記コンピュータが前記保護コード実行形式を実行する前に前記中央演算装置に投入され、前記暗号手段によって前記秘密
- 15   鍵を用いて復号され、前記保護コード実行形式は、前記ライセンスから取得された前記コード暗号鍵を用いて、前記暗号手段によって復号される、ことを特徴とする。

- このプログラム生成装置によって、上記中央演算装置によって保護を受けることができない形式のコードモジュールから、上記中央演算装置によって保護
- 20   を受けることが可能なプログラムを生成することが可能となる。生成されたプログラムは、上記中央演算処理装置を有するコンピュータによって実行される事によって、ハードウェアTRM並みの安全性を得ることができる。

#### 図面の簡単な説明

- 25      図1は、基本パッケージソフトウェアの利用関係モデルを示す図である。

図 2 は、プログラミングモデルを示す図である。

図 3 は、第 1 実施形態に係わる G T を実現する C P U の構成図である。

図 4 は、T R B 内の各行のフィールドの構成を示す図である。

図 5 は、T L B 内の各行の拡張フィールドの構成を示す図である。

- 5 図 6 は、F R アーキテクチャの場合について、T L B 内のフィールドの値とアクセス権限の対応を示す図である。

図 7 は、インテルアーキテクチャの場合について、T L B 内のフィールドの値とアクセス権限の対応を示す図である。

図 8 は、暗号化キーテーブルの構造を示す図である。

- 10 図 9 は、署名方式の一例を示す図である。

図 1 0 は、T R B、T L B、暗号化キーテーブル及びページテーブルの関係を  
示す図である。

図 1 1 は、T R S S フラグを格納するフィールドの構成の一例を示す図である。

- 15 図 1 2 は、R A C T の各行のフィールド構成の一例を示す図である。

図 1 3 は、R S S L レジスタのフィールド構成の一例を示す図である。

図 1 4 は、一般の仮想セグメント空間と耐攻撃セグメント空間との間でコード  
実行プロセスからデータにアクセスする際のポリシーを示す図である。

図 1 5 は、G T 1 0 上で保護プロセスが動作する様子を示す図である。

- 20 図 1 6 は、G T の認証を説明する図である。

図 1 7 は、G T ライセンスに設定可能なアクセス権と被許諾権限を示す図である。

図 1 8 は、上述の G T 認証処理の一部を示すフローチャートである。

- 25 図 1 9 は、超流通ファイル形式として用いる場合の保護コード実行形式の一例を示す図である。

図 2 0 は、保護コードのロードと実行手順、及び保護データの退避と維持について説明する図である。

図 2 1 は、保護コードを実行する際の保護コードを記録するページへのアクセス制御について説明する図である。

- 5 図 2 2 は、プロセッサコアによって行われる保護コード及び保護データへのアクセス制御を示すフローチャートである。

図 2 3 は、例外・割り込み処理にかかわるフローチャートである。

図 2 4 は、命令処理にかかわるフローチャートである。

- 図 2 5 は、保護コードファイルを起動する際における、OS カーネル 6 0 及び G T の動作について説明する図である。

図 2 6 は、保護コードを実行するプロセスから保護データをアクセスするメカニズムについて説明する図である。

図 2 7 は、親プロセスから他の保護コードモジュールを呼び出す際に行う手順について説明する図である。

- 15 図 2 8 は、親プロセスから他の保護コードモジュールを呼び出す際に OS カーネルが行う処理のフローチャートを示す。

図 2 9 は、耐攻撃コード復帰例外処理による封印レジスタ不正アクセス防止のためのシーケンス例を示す図である。

- 図 3 0 は、OS カーネルによる保護コンテキスト切り替え時のシーケンスの一例を示す図である。

図 3 1 は、保護データ領域の共有の一例を示す図である。

図 3 2 は、モジュールの認証、並びに、保護データ復号鍵共有手順の設定手順について説明する図である。

- 図 3 3 は、耐攻撃領域共有システムコールを呼び出した際の処理を示すフローチャートである。



活させる時にそのデータを復号する事としているため、このような問題を回避する事が可能となる。

なお、キャッシュ 1 2 及び 1 3 と RAM 1 7 の間に、平文のコードブロック用のキャッシュライン、平文のデータブロック用のキャッシュライン、保護コードブロックを復号するためのキャッシュライン、保護データブロックを暗号化するためのキャッシュライン及び、保護データブロックを復号するためのキャッシュラインのように、複数のキャッシュライン並列に備える事としても良い。これにより、GT 1 0 による処理を高速化し、保護コードブロック及び保護データブロックの安全性を高めることが可能となる。

10     また、図 3 に示すように、プロセッサコア 1 1 内には、レジスタが備えられる。GT 1 0 によれば、従来の仮想メモリ領域に、耐攻撃領域という機構を追加し、この耐攻撃領域内では、安全上のリスクが互いに影響することなく、複数の保護ソフトウェアを実行する事を可能としている。そのために、プロセッサコア 1 1 は、耐攻撃セグメントセクタ（以下、TRSS という）フラグ、  
15     レジスタアクセスコントロールテーブル（以下、RACT という）及びレジスタ封印状態リストレジスタ（以下、RSSL レジスタという）を備える（不図示）。

TRSS フラグは、仮想メモリ領域内のセグメントを指定するセグメント指定レジスタ内のフィールドに記録される。プロセッサコア 1 1 は、TRSS フラグに基づいて、レジスタによって示される仮想アドレスが、耐攻撃セグメント空間内であるのか、一般の仮想セグメント空間内であるのかを判断する事ができる。また、複数の保護ソフトウェアを実行する際に、現在実行中のプロセス以外のプロセスからレジスタにアクセスすることができないように、プロセッサコア 1 1 は、RACT を用いて、レジスタの封印及び解放を制御する。また、  
25     さらに、レジスタが封印されているのか解放されているのかを示す情報は、

RSSLレジスタに登録される。耐攻撃セグメント空間及び耐攻撃領域並びにレジスタの封印及び解放について詳しくは後述する。

このように、GT10は、TRM化した1ロットのCPUを用いて実現される。そして、GT10は、GT10上で動作する、DRMソフトウェアモジュール又はその他の保護を要するモジュールをTRM化することができる。従って、ハードウェアTRMを製造するコスト、特に初期ロットの開発コストを削減する事が可能となる。

以下、TRB、TLB、TRSSLレジスタ、RACT及びRSSLレジスタについて順に説明する。

10 TRBは、プログラムコード（インストラクションの連結）を復号するための鍵 Kc 及びデータを暗号化／復号化する鍵 Kd 等を保持する。なお、Kd 及び Kc を合わせてソフトウェア暗号鍵という事もある。TRBは、カーネルモードであってもユーザが内部を参照・改ざんすることができない構造、すなわち耐攻撃構造をもつ。TRB内で Kc や Kd を保持する位置の識別は Key ID によつて行われ、TLB内にもリンクした鍵の位置を識別する Key ID を持つ。この

15 Key IDを用いて、TRBとTLBをリンクさせる。

図4に、TRB内の各行のフィールドの構成表を示す。図4の表に示すように、TRB内の各行は、有効性フラグ p、外部出力禁止フラグ uo、キャッシュロックフラグ cl、鍵識別情報 kid、暗号鍵 key、被許諾コード鍵 ackey 及び例

20 外処理アドレス eadr 等のフィールドを含む。また、これらのフィールドのサイズは、例としてあげたものであり、GT10のアーキテクチャや用途によって他の最適な値にすることが可能である。

有効性フラグ p は、TRBが有効であるか無効であることを示すフラグであり、オン（1）である場合、TRBが有効であることを示す。

25 外部出力禁止フラグ uo は、その行の内容を暗号化キーテーブルに出力しても

良いか否かを示すフラグであり、オン（１）である場合、暗号化キーテーブルに出力されない。但し、この場合でも、管理情報として必要な p、uo、cl 及び kid は、出力可能としてもよい。外部出力禁止フラグ uo のデフォルト値は、オフ（０）であり、その場合はその行の内容は暗号化キーテーブルに出力される。

- 5 この場合、TRB 内の内容と暗号化キーテーブル内の内容とを同期させる必要がある。なお、暗号化キーテーブルは、GT10 内の TRB の内容を格納する格納手段である。暗号化キーテーブルについて詳しくは後述する。

- キャッシュロックフラグ cl は、鍵識別情報 kid によって指定された耐攻撃領域内のデータがキャッシュの外に出力されるか否かを示し、オン（１）である  
10 場合、そのデータ又はコードはキャッシュの外に出力されない。なお、cl がオン（１）である場合、次の２つのモードを切り替える機能を更に GT10 に備える事としても良い。

（a）CPU 内蔵キャッシュまでしか保護データを出さないモード

（b）外部（三次）キャッシュまで保護データを平文で記録するモード

- 15 （a）は保護データを記録する領域を少なくし、暗号・復号処理をしないことで処理性能を上げる必要がある場合にも利用することができる。（b）は処理性能向上を期待できるが、保護強度のレベルは落ちることとなる。

鍵識別情報 kid は、鍵を識別する情報であり、TLB と TRB をリンクさせるために用いられる。

- 20 暗号鍵 key は、そのラインにリンクするページに書き込まれたコード又はデータを暗号化・復号するための暗号・復号鍵の値である。暗号鍵は、例えば、対称鍵（共通鍵）暗号法の鍵である。このフィールドに Kc や Kd が記録される。

- 被許諾コード鍵 ackey は、リンクするページにアクセス可能な保護プロセスの実行コードを復号するための復号鍵である。ここで、保護プロセスとは、  
25 保護コードが実行された状態をいう。その実行コードのページは、TRB 内の

5 行の鍵識別情報 kid と同じ kid を含む T L B 中の行で、ページ番号を用いて指  
 定され、そのページへのアクセス権の種類も T L B 中の行で指定される。通常、  
 被許諾コード鍵 ackey は、他の行及び自行の暗号鍵 key である。ただし、  
 ackey の全ビットが ' 0 ' であれば、すべてのプロセスが、G T ライセンスを用  
 いてアクセス権を許諾されたことを示すものとする。また、ACgt. ackey (ACgt

10 例外処理アドレス eadr は、key が異なるページからこの T R B 内の行にリン  
 クしたページに復帰する直前に発生する例外処理コードの先頭アドレスを示す。  
 例外処理アドレス (eadr) は、ACgt. eadr (ACgt 内の eadr フィールドの値) が含  
 まれない AUTHORIZE 命令により TRB 内に行が設定された時点では、デフォルト  
 値として全ビットに ' 0 ' が設定される。保護プロセスへの復帰時等に、「耐攻  
 撃コード復帰例外アドレス不正例外」が発生した際には、そのプロセスの実行  
 15 を停止し、当該 TRB の有効性フラグ (p) を off にしなければならない。なお、  
 将来、データページごとにアクセス例外を設ける必要がある場合には、  
 TRB. eadr (T R B 内の eadr フィールドの値) は、TRB. ackey (TRB 内の ackey  
 フィールドの値) で暗号化されたコードの実行時に実行する例外処理コードの  
 アドレスとして定義することもできる。

20 鍵 key フィールドに格納されるコード復号鍵 Kc 及びデータ暗号化・復号鍵  
 Kd は、例えば、乱数として生成する対称鍵暗号方式の暗号鍵であり、G T 1 0  
 において、暗号・復号の前と後でデータ長が同じになる暗号方式を選択するこ  
 ととしてもよい。鍵を乱数生成アルゴリズムを用いて生成しても良いし、G T  
 1 0 内の熱乱数発生機能などを用いて生成してもよい。前者の生成法より後者  
 25 の生成法の方が安全である場合が多い。Kc 及び Kd は、後述の G T ライセンス

に含まれる。コード復号鍵 Kc 又はデータ復号鍵 Kd と、アクセス権等を埋め込んだ G T ライセンスをパラメタとして C P U インストラクション「アクセス許可命令 (AUTHORIZE 命令)」(後述)を実行する事により、耐攻撃領域内の T L B (後述)にリンクされた T R B 行内の各フィールドに値が設定される。

- 5     T L B は、保護コード及び保護データを格納するアドレス並びにページへのアクセス権を管理する。図 5 に、T L B 内の各行の拡張フィールドの構成表を示す。図 5 に示すように、G T 1 0 に係わる T L B 内の各行は、従来の T L B が持つフィールドに加え、有効性フラグ p、領域識別子 id、物理ページ番号 ppn、仮想ページ番号 vpn、アクセス権限 rights、鍵識別情報 kid、暗号化ブロック識別方式 ebim 及びデジタル署名 sign 等のフィールドを含む。なお、アクセス権限 rights フィールドは、権限レベル pl、アクセス権 ar、耐攻撃性フラグ tr、デバッグモードフラグ df に分けられる。これらのすべてのフィールドは G T 1 0 内ではこの耐攻撃領域内になければならない。また、図 5 において示されたフィールドのサイズは、例としてあげたものであり、G T 1 0 のアー
- 10     キテクチャや用途によって他の最適な値にすることが可能である。

有効性フラグ p は、T L B が有効であることを示す。

領域識別子 id は、T L B 内の当該行が示すページ領域の識別子である。

物理ページ番号 ppn は、物理ページ番号を示す。仮想ページ番号 vpn は、仮想ページ番号を示す。

- 20     アクセス権限 rights は、その当該行が示すページへのアクセス権限を示す。権限レベル pl はページにアクセス可能な権限のレベルを示し、アクセス権 ar は、ページへのアクセス権の種類を示す。権限レベル pl 及びアクセス権 ar は、図 6 及び図 7 に示すようにして T L B の各フィールドの値に基づいて決定され、T L B に設定される。なお、図 6 は、F R アーキテクチャの場合を示し、図 7
- 25     は、インテルアーキテクチャの場合を示す。

耐攻撃性フラグ *tr* は、当該行が示すページが耐攻撃セグメント空間内（後述）内にあるか否かを示し、耐攻撃性フラグ *tr* がオン（１）であれば、そのページは耐攻撃セグメント空間内にあり、その行に対応する行が *T R B* 内にあることを示す。耐攻撃性フラグのオン・オフは *G T 1 0* の認証時に行われる。

- 5     デバッグモードフラグ *df* は、デバッグモードが有効であるのか無効であることを示す。デバッグモードフラグ *df* がオン（１）であれば、有効である。耐攻撃性フラグ *tr* とデバッグモードフラグ *df* がオン（１）である場合、アクセス権 *ar* で指定された値に応じたデバッグモードが動作し、各 *ar* の値の意味は以下ようになる。

- 10     (a) *P R* の場合、全プロセスから読み出しのみ可能： *R*  
        (b) *X* の場合、全プロセスからの読み出しと実行が可能： *R X*  
        (c) *P R W* の場合、全プロセスからの読み出しと書き込みが可能： *R W*  
        (d) *P W X* の場合、全プロセスからの読み書きと実行が可能： *R W X*

- 鍵識別情報 *kid* は、*T R B* 内の鍵情報にリンクするために *T R B* 内の行  
 15     (insertion) を識別する情報である。

       暗号化ブロック識別方式 *ebim* は、暗号化されているコードブロック又はデータブロックを識別する情報である。

- a) *ebim*=0 の場合、ブロックは平文である。  
        b) *ebim*=1 の場合、ブロックは暗文である。（デフォルト値であり、*ebim*  
 20     フィールドが省略されている場合、*ebim*=1 として扱われる）

       デジタル署名 *sign* は、上述の *vpn* から *ebim* までのフィールドを連結したフィールド連結データのデジタル署名であり、*G T 1 0* が生成する。

- 保護コード及び保護データが格納される領域を示す *T L B* インサクション（行）内ラインのアクセス権限の値は、後述の *TLB.tr* がオフの場合に限り、  
 25     OS が管理する。従来通り、本発明でもアクセス権限の値は 3 ビット程度で表

現されるが、本発明によれば、さらにページが「耐攻撃領域内にあることを示すフィールドとして「耐攻撃性フラグ tr」フィールドをTLBに設ける。また、この耐攻撃領域を利用可能なコード実行状態を「耐攻撃モード(Tamper Resistant Mode)」と呼ぶものとする。

- 5      保護コードおよび保護データを格納するアドレスはそのページを利用する前にTLBに設定される。アクセス権限 rights 及び暗号化ブロック識別方式 ebim、被許諾コード鍵 ackey は、GTライセンスに含まれるGTアクセス条件 ACgt（後述）に基づいて、決定される。

- コード復号鍵 Kc 又はデータ暗号・復号鍵 Kd と、アクセス権とを埋め込んだ  
10    GTライセンスをパラメタとしてCPUインストラクション「アクセス許諾命令」（後述）を実行する事により、TLB行内に、保護ページごとに KeyID が指定され、各 KeyID に対応するTRB行内の key フィールドに復号鍵が設定される。

以下、TRB及びTLBが格納される場所について説明する。

- 15    TRBは、GT10内に格納されるが、GT10に備えられた記憶容量が一杯になる場合が想定される。この場合、GT10内のTRBの内容の少なくとも一部を暗号化し、暗号化された内容をRAM17やハードディスク等の外部記憶装置に記録する。この外部に記録されたTRB内の行のリストを暗号化キーテーブルという。そして、GT10は、TRBの内容を暗号化した状態で外部  
20    記憶装置内の暗号化キーテーブルに記録しながら管理し、必要な情報を暗号化キーテーブルから取り出してGT10内で復号して利用する。TRBの内容を暗号化したり復号したりする際に用いられる鍵 Ktrb は、例えば、対称鍵暗号法の秘密鍵である。この暗号鍵 Ktrb は、GT10の入力電源が落ちたり、GT10がリセットされたりした場合であっても、継続して維持されることが  
25    望ましい場合がある。

以下、図 8 を用いて暗号化キーテーブルの構造について説明する。図 8 に示すように、暗号化キーテーブルには、有効性フラグ  $p$ 、外部出力禁止フラグ  $uo$ 、鍵識別情報  $kid$  及び暗号化された  $TRB$  の内容を格納するフィールド等が含まれる。

- 5     図 8 に示すように、暗号化キーテーブルに記録する際、各行にヘッダとして平文のままの  $TRB.p$  ( $TRB$  中の有効性フラグ  $p$  フィールドの値) と平文のままの  $TRB.kid$  ( $TRB$  中の鍵識別情報  $kid$  フィールドの値) を付加する。これにより、 $OS$  (スーパーバイザモード) によって暗号化キーテーブルを管理することが可能となるが、 $Key ID$  以外の暗号化されている内容は参照も改ざんも
- 10    できないこととなる。 $TRB$  内のフィールドのうち、 $p$  と  $kid$  とは平文の状態で特定レジスタなどからみえてもよい。

- $TRB$  の内容を外部記録装置に記録する際には、他の保護データブロックと同様にデータキャッシュ 13 の暗号化ラインを通して行う。また、図 8 に示すように、 $TRB$  の各行を暗号化する際には行の先頭に乱数を付加してから暗号
- 15    化しなければならない。これにより、 $Kc$  (コードの暗号鍵) を指定したソフトウェア開発者が故意に  $TRB$  中の暗号鍵  $key$  フィールドの値を何度も入れなおして、平文と暗文のペアを大量に生成することにより、容易に  $TRB$  を暗号化する暗号鍵  $Ktrb$  を解読することを防止する。

- さらに、暗号化キーテーブルを  $RAM 17$  から不図示のハードディスクに記
- 20    録しておき、 $GT 10$  の再起動後に再利用することとしてもよい。特にハードウェア  $TRM$  レベルの強度をもつソフトウェア  $DRM$  を構築するために暗号鍵  $Ktrb$  で暗号化されたままの  $Kd$  (データ暗号化鍵) をハードディスクやフラッシュメモリなどに保存し管理する必要がある場合もあることから、 $TRB$  の暗号鍵  $Ktrb$  は、 $GT 10$  の電源切断後も耐攻撃領域内で  $F e R A M$
- 25    (Ferroelectric Random Access Memory) 等の不揮発性メモリなどに維持し続



けてもよい。

暗号化キーテーブルと T R B の内容の同期方式にはつぎのような方式が考えられるが、本発明がそのいずれかに限定されるものではない。

- (a) 暗号化キーテーブルへの T R B の内容の書き出しと暗号化キーテーブルから T R B への再取り込みは特定のレジスタへの R W 命令などを介して行う。

(b) 暗号化キーテーブルの先頭アドレスと行数を設定する（特別なインストラクションを実行する）ことにより、必要になった際に G T 1 0 が自動的にテーブルと T R B の同期をとる。また、ユーザが必要な時に同期がとれるようにフラッシュインストラクションを G T 1 0 に実装する。

- 10 一方、T L B の内容は、一般の C P U と同様に G T 1 0 でも、外部記憶装置にページテーブルとして記録し、G T 内の T L B の内容とページテーブルの内容とは同期させて管理される。そして、G T 1 0 は、必要な情報を必要な時にページテーブルから G T 1 0 内に取り込んで利用することとしてもよい。

以下、T L B とページテーブルの間での内容の受け渡しについて説明する。

- 15 T L B の内容をページテーブルに記録させる際には、TLB. tr (T L B 内の tr フィールドの値) がオンの場合、G T 1 0 は、TLB. sign (T L B 内の署名 sign フィールドの値) を生成し、記録させたい内容につける。そして、ページテーブル内の内容を G T 内の T L B に取り込む際には、G T 1 0 は、内容に付されている署名を確認し、署名が正しくなければ、T L B 署名不正例外（後述）を  
20 発生させ、その行の有効性フラグ (TLB. p) と耐攻撃性フラグ (TLB. tr) を無効（オフ）にする。

- 図 9 に署名方式の一例を示す。図 9 に示すように、まず、G T 1 0 は、TLB. vpn、TLB. rights、TLB. kid 及び TLB. ebim を含むデータに G T 1 0 内の固定の乱数を連結する。続いて、連結されたデータを SHA-1 でハッシュし、G T  
25 1 0 内の秘密鍵 Ktlb を用いて暗号化する。これにより署名 TLB. sign を生成す

る。署名を付す対象となる内容が 160 ビットに満たない場合、例えば埋め草としての乱数フィールドをその内容に追加する事としてもよい。

以下、図 10 を用いて、TRB、TLB、暗号化キーテーブル及びページテーブルの関係について説明する。

- 5     図 10 に示すように、TRB 及び TLB は、GT10 内に保持される。TRB には、鍵識別情報 kid、コードブロックを復号する際に用いられる Kc 及びデータブロックを暗号化・復号する際に用いられる Kd 等が記録される。TLB には、領域識別子 id、鍵識別情報 kid、仮想ページ番号 vpn、物理ページ番号 ppn、アクセス権限 rights 等が格納される。領域識別子 id は、コードブロッ  
10   クやデータブロックが格納されている RAM17 内のページに対応する。また、TRB と TLB の内容は、鍵識別情報 kid によって対応付けられている。

- TRB の内容を暗号化キーテーブルに記録する場合、その内容に乱数を追加してから暗号鍵 Ktrb を用いて暗号化する。暗号化キーテーブルの内容を GT10 内に取り込む場合、暗号鍵 Ktrb を用いてその内容を復号する。一方、T  
15   LB の内容をページテーブルに記録する場合、GT10 は、暗号鍵 Ktlb を用いてその内容に基づく署名 TLB.sign を生成し、その内容に追加する。

- 続いて、GT10 内のキャッシュと、TLB と、TRB との同期について説明する。上記のように、TLB や TRB には、ページに記録された保護コードや保護データへのアクセス制御に関する情報が記録されている。そこで、同じ  
20   仮想アドレスをもつ 2 つの TLB をハードウェアまたはソフトウェアで偽装することにより、保護コード又は保護データにアクセスしようとする攻撃が考え得る。このような攻撃に対処するため、GT10 において、プロセッサコア 11 は、キャッシュと TLB と TRB の内容の同期を取りながら、アクセス制御の判断を実施することとしてもよい。より具体的には、命令キャッシュ 12 内  
25   の保護コード又はデータキャッシュ 13 内の保護データの内容は、TLB 行の

仮想アドレス (TLB.vpn) だけでなく、その T L B 行のデジタル署名 (TLB.sign) の値と仮想アドレスのペアにリンクすることとしてもよい。この場合、プロセッサコア 1 1 は、このペアが不一致の際には異なるページとしてアクセス制御の判断をおこなう。

- 5      このように、GT ライセンスにアクセス許可保護コードのコード復号鍵(Key) を被許諾コード鍵(Authorized Code Key)として埋め込んでおくことで、保護領域にアクセス可能な保護コードを指定することができる。そして、T R Bの Authorized Code Key フィールド(ackey)で指定された Key をもつ仮想ページ上のコードを実行したコードからのみ保護コード及び保護データにアクセス可能
- 10     とする。ただし、アクセス権が X または P W X のコードを実行する場合に限り、そのページの ackey の値がすべて 0 であれば、すべてのコードからの実行を許諾するものとする。

- ページ A の ackey として指定された鍵に一致する key の値をページ B が持つとき、ページ B 上の保護コードの実行状態をここではページ A (上のコード／
- 15     データ) の「オーナープロセス (Owner Process)」と呼ぶ。

なお、G T における実行権 (X) とは、TRB.ackey (T R B の ackey フィールド) に値を持つ耐攻撃ページ上の命令コードを実行する権利を意味する。この権利はその命令コードのひとつ手前で実行された命令コードに対して与えられるもので、つぎのようなケースがある。

- 20     (a) ひとつ手前で実行された命令のあとに現在実行すべき命令とが続いている場合。

(b) ひとつ手前で実行された命令が CALL や JUMP などの分岐命令である場合。

- 特に前の命令と次の命令の仮想ページが異なる場合、G T 1 0 は、あらため
- 25     て次に実行することを指定された命令コードが実行を許諾されているかどうか

を確認しなければならない。この確認については後述する。

続いてTRSSフラグについて説明する。図11に、TRSSフラグを格納するフィールドの構成の一例を示す。GT10はその仮想メモリ空間のセグメント指定レジスタに、TRSSフラグを持つ。このフラグはコード・セグメント、データ・セグメント、スタック・セグメントのそれぞれに対して存在し、それぞれ耐攻撃コードセグメントセクタ (TRCSS: Tamper Resistant Code Segment Selector)、耐攻撃データセグメントセクタ (TRDSS: Tamper Resistant Data Segment Selector)、耐攻撃スタックセグメントセクタ (TRSSS: Tamper Resistant Stack Segment Selector)と呼ぶ。TRSSフラグがオンである場合、ページは耐攻撃空間内に設定され、オフである場合、ページは一般の仮想空間内に設定される。

続いて、RACTについて説明する。図12に、RACTの各行のフィールド構成の一例を示す。GT10によれば、レジスタの封印と解放の機能を実現するために、耐攻撃モジュール内にRACTを備える。RACTの各行は、レジスタIDrid、封印フラグ seal 及び被許諾コード鍵 ackey 等のフィールドを含む。

レジスタIDrid は、レジスタを指定するIDである。封印フラグ seal、レジスタが封印中であるか否かを示し、オン (1) である場合レジスタは封印中であり、オフ (0) である場合レジスタは解放されている事を示す。非許諾コード鍵 ackey は、TRBの ackey と同様であり、レジスタへのアクセスが許可されているプロセスのコードページの鍵 key である。

続いて、RSSLレジスタについて説明する。RSSLレジスタは、GTに必須の機能ではないが、オプション機能として備えることとしてもよい。RSSLレジスタは、各レジスタの封印状態を示す情報を格納する。図13にRSSLレジスタのフィールド構成の一例を示す。図13に示すように、RSSL

レジスタは封印可能なレジスタの数と同じビット長を有し、各ビットは、そのビットに対応するレジスタの封印状態を示す。ビットがオン（1）である場合、当該レジスタが封印されていることを示し、オフ（0）である場合、当該レジスタが解放されていることを示す。例えば、RSSLレジスタの第1ビットが  
 5 レジスタ r 1 の封印状態を示すように設定されている場合に、第1ビットがオンであれば、レジスタ r 1 は封印されていることを意味する。

続いて、耐攻撃セグメント空間及び耐攻撃領域について説明する。上述のように、TLB内の耐攻撃性フラグ tr がオン（1）であるページは、耐攻撃セグメント空間内に含まれる耐攻撃領域である。また、このページに対応するセ  
 10 グメント指定レジスタ内の行において、TRSSフラグはオンである。耐攻撃領域内のページには、保護データが格納される。図14に、一般の仮想セグメント空間と耐攻撃セグメント空間との間でコード実行プロセスからデータにアクセスする際のポリシーを示す。一般仮想セグメント空間は、保護を必要がないデータ及びコードが記録される空間であり、耐攻撃セグメント空間は保  
 15 護データ及び保護コードが記録される空間である。

耐攻撃セグメント空間で実行されるコードから一般仮想セグメント空間内のデータにアクセスすることはできるが、一般仮想セグメント空間で実行されるプロセスから耐攻撃セグメント空間内のデータにアクセスはできない。また、ユーザレベル空間とスーパーバイザレベル空間とのアクセス制御関係は、耐攻  
 20 撃セグメント空間内でもライセンスオーナーが同じであれば、一般仮想空間における場合と同じポリシーが維持される。

また、耐攻撃空間内であっても、ライセンスオーナーが異なる保護コード及び保護データの領域間では、相互にアクセスすることができない。ただし、後述の耐攻撃領域共有機能を用いることで、相互アクセスを可能にすることがで  
 25 きる。

上述のアクセスポリシーにより、GT10における保護プロセスは、他のプロセスからの影響を受けない。GT10において、保護プロセスは自身が生成した耐攻撃ページを持ち、一般にOSによって管理される。

図15に、GT10上で保護プロセスが動作する様子を示す。図15に示すように、GT10上では、複数の保護プロセスが動作し、それぞれの保護プロセスは耐攻撃領域内にページを生成する。作業データは、GT10内の暗号エンジン16によって暗号化されてから耐攻撃領域（仮想メモリ領域）に格納される。従って、仮想ハードウェアTRMの範囲は、RAM17やハードディスク等までも伸び縮みし、一定の形を持たないといえる。このため、GT10が生成し動作する保護プロセスは「仮想ハードウェアTRM（Virtual Hardware Tamper Resistant Module :VHTRM）」内で動作しているといえることができる。故に、GT10によれば、ハードウェアTRM並みの強度を持ちつつも、ハードウェアTRMに伴うリソースの制限という問題を解消することが可能となる。

また、保護プロセスは、世界中のCPU（GT10）空間を自在に移動することも可能である（後述のGRID計算）。このように、どのような攻撃にも耐え、自分の使命を果たして元のGT10に帰ってくるVHTRMを纏った保護プロセスを擬人化して「耐攻撃エージェント(Tamper Resistant Agent)」と呼ぶこともできる。

#### <インストラクションセット>

次に、GT10に与えられるインストラクションセットについて説明する。

まず、GTを実装するCPUが、従来のCPUのインストラクションセットに加えて、GT10としての機能を実現するために備える基本命令について説明する。基本命令には、以下の（1）から（9）がある。以下、各命令について順に説明する。

（1）GT証明書取り出し命令：Store GT certificate command

命令形式：STR\_GT\_CERT (certNum, certAdr)

入力レジスタ：

certNum：G T 1 0 内でローカルに割り当てられた証明書番号。N 枚の証明書があれば、0 から N-1 までの値が有効となる。

5 certAdr：証明書の内容を書き込むアドレス

出力（例外処理）：

errorCode：指定した番号の証明書がない場合などに設定される。

動作：certNum で指定された証明書を G T 1 0 内から取り出し、certAdr で指定されたアドレスに書き出す。

10 動作可能権限：全レベル

（2）アクセス許諾命令：Access authorization command

命令形式：AUTHORIZE (licenseAdr, resultAdr, startVPN, numOfVP, kid)

入力レジスタ：

licenseAdr：G T ライセンス（後述）を格納したアドレス

15 resultAdr：結果を格納するアドレス

startVPN：耐攻撃領域の先頭仮想ページ番号

numOfVP：連続する耐攻撃領域のページ数

kid：OS が T L B と T R B の関係を管理するために割り当てる識別子

出力：

20 result：[G T ライセンス || startVPN || numOfVP] || 以上の署名] が resultAdr に記録される。

errorCode（例外処理）：認証が失敗した場合などに設定される。

動作：指定レジスタのメモリ上にある G T ライセンスを認証し、G T ライセンスから取り出した耐攻撃領域のコード又はデータ復号鍵 key（Kc 又は Kd）、

25 kid、ackey を耐攻撃領域内の T L B にリンクする T R B に設定する。さらに、

- T L Bに「耐攻撃フラグ」(tr)とアクセス権 (PR, X, PRW, PWX) 、ebim、kid  
及び sign を設定する。正しく設定が完了すれば、入力データを連結してその  
デジタル署名を付加し、resultAdr で指定されたアドレスに記録する。このと  
き、入力データの連結をハッシュし、結果を Kgt を用いて暗号化し、その結果  
5 をデジタル署名として採用する。認証が失敗した場合やT L Bが存在しない場  
合など、設定が失敗した場合には例外を発生させる。

動作可能権限：スーパーバイザレベル（権限レベル0）のみ

（3）耐攻撃権限設定命令：Set Tamper Resistant Rights command

命令形式：SET\_TR\_RIGHTS (rights, startVPN, numOfVP)

- 10 入力レジスタ：

rights：アクセス権 (ACgt.ar (後述) を指定)

startVPN：耐攻撃領域の先頭仮想ページ番号

numOfVP：耐攻撃領域のページ数

出力（例外処理）：

- 15 errorCode：入力に対応する行がT L B又はT R Bにない場合、又はすで  
に設定されているアクセス権よりも指定されたアクセス権の範囲が広い場合な  
どに発生。

- 動作：startVP 及び NnumOfVP で指定された仮想領域が耐攻撃空間である場合、  
その領域のT L Bに、rights で指定されたアクセス権を設定する。ただし、  
20 rights で指定されたアクセス権は、すでにT L Bに設定されているアクセス権  
の範囲でなければならない。

動作可能権限：スーパーバイザレベル（権限レベル0）のみ

（4）耐攻撃例外アドレス設定命令：Set Tamper Resistant Exception  
command

- 25 命令形式：SET\_TR\_EXCEPTION (startAdr)



入力（イミディエイトまたは直接アドレッシングのみ）：

startAdr：T R Bに設定する耐攻撃空間内の仮想アドレス。レジスタ解除・復帰例外などのジャンプ先として指定する。なお、封印したいレジスタの使用前に startAdr を秘匿して実行する必要があることから、レジスタは入力

5 パラメタとして利用できない。

出力（例外処理）：

errorCode：startAdr で指定されたアドレスに対応する行がT R B内に存在しない（耐攻撃モードではない）。又はそのアドレスへのアクセス権がない。

動作：指定された startAdr を現在実行中のコードページのT R B. eadr（T  
10 R B内の eadr フィールド）に設定する。

動作可能権限：全レベル

（5）レジスタ封印命令：Seal Register command

命令形式：SEAL\_REG (registerList)

入力レジスタ： registerList：封印するレジスタ一覧（フラグリスト）。

15 各レジスタに対応したフラグについて、オン（1）でレジスタを封印することを示し、オフ（0）で封印しないことを示す。

出力（例外処理）：

errorCode：指定したレジスタが存在しない。または、そのレジスタはすでに封印済である。

20 動作：指定したレジスタには、本命令を出したプロセス（コード実行状態）コードページの暗号鍵 Kc と同じ鍵を持つコードページのプロセス以外からアクセスできなくなる。本命令の実行によって、対応するレジスタの RACT. ackey（RACT の ackey フィールド）に、本命令を実行したコードを暗号化した key の値を記録する。

25 動作可能権限：全レベル

(6) レジスタ解放命令 : Release Register command

命令形式 : RELEASE\_REG (registerList)

入力レジスタ :

registerList : 封印を解放するレジスタ一覧

5 出力 (例外処理) :

errorCode : 指定したレジスタが存在しない。または、そのレジスタは解放済である。

動作 : 指定した範囲のレジスタの値をすべて 0 にクリアし、封印を解除する。

動作可能権限 : 全レベル

10 (7) 非許諾領域スタックストア命令 : Store Stack to Unauthorized Area command

命令形式 : STMEA\_TO\_UA (registerList)

入力 :

registerList : スタックに記録されるレジスタの一覧。フラグのリスト。

15 各レジスタに対応したフラグがオン (1) の場合、そのレジスタをスタックに記録することを示し、オフ (0) で記録しないことを示す。

出力 (例外処理) :

errorCode : registerList で指定されたレジスタが封印されていない。

20 動作 : スタックポインタが、許諾されていない耐攻撃領域を示す場合でも、指定されたレジスタの RACT. ackey (RACT 内の ackey フィールド) の値がその領域に対応する TRB. ackey (TRB 内の ackey フィールド) に一致すれば、指定されたレジスタがスタックポインタの位置に記録される。

動作可能権限 : 全レベル

25 (8) 非許諾領域スタックロード命令 : Load Stack from Unauthorized Area command

命令形式：LDMEA\_FROM\_UA (registerList)

入力：

registerList：スタックから読み出すレジスタ一覧

出力（例外処理）：

- 5        errorCode：指定レジスタが封印されている。

動作：スタックポインタが耐攻撃領域を示す場合、指定されたレジスタの RACT. ackey にスタックポインタの領域の T R B. ackey の値がコピーされ、指定されたレジスタの RACT. seal (R A C T内の seal フィールド) がオン (1) に設定される。さらに、指定されたレジスタにスタックポインタの位置のデータがロードされる。

10

動作可能権限：全レベル

(9) ライセンス削除命令：Delete License commnad

命令形式：DELETE\_LICENSE(kid)

入力：

- 15        kid：OS が T L B と T R B の関係を管理するために割り当てた識別子で、AUTHORIZE 命令のパラメタとして用いた値。

出力（例外処理）：

errorCode：kid が誤っている場合、又は削除に失敗した場合等に設定される。

- 20        動作：指定された kid の T R B 及びそれにリンクするすべての T L B が削除され、対応するページテーブルと暗号化キーテーブルの有効性フラグもオフに設定される。また、それらの T L B が示していた仮想領域のキャッシュロックも解除される。

動作可能権限：スーパーバイザレベル（権限 0 レベル）のみ

- 25        上述の (1) から (9) までの基本命令に加えて、性能や機能の向上のため

の拡張命令として、(10) から (15) までの命令等が考えられる。以下、順に拡張命令について説明する。

(10) 保護ブロック開始命令 : Protected Block Start command

命令形式 : PRT\_BLOCK\_START (encryption\_flag)

5 入力 (直接パラメタ) :

encryption\_flag : オン (1) ならブロックが暗号化されていることを示す。

動作 :

命令データの特定のビット位置が特定パターンにセットされていれば暗号  
10 化ブロック開始命令とする。本命令内の特定ビット列 (7ビット程度) は暗  
号化ブロックの長さを指定するために用いられ、値を指定する単位は、キャ  
ッシュ可能なインストラクション長の最小値と同じとする。また指定可能な  
最大値はキャッシュ可能な長さの最大値と同じとする。本命令のその他のバ  
イトには乱数を入れなければならない。本命令データの先頭の境界は命令キ  
15 ャッシュ単位の境界に一致しなければならない。

乱数の長さがセキュリティ上十分でない場合は、この命令を2つ以上連続  
する。連続した場合、最後の本命令以外はブロック長の値に0が入る。

保護コードブロック開始命令は、GT10内で復号された後、同じ長さの  
Nop 命令となる。

20 TLB.ebim の第2ビットがオンの場合、ブロックに含まれるハッシュまたは  
署名 (平文の場合) をチェックしなければならない。ハッシュまたは署名の  
位置は、例えばブロックの末尾とする。ブロックがハッシュや署名を含まな  
い場合は、ハッシュ非実装例外を発生させる。

なお、この命令は、第2実施形態 (後述) に対応する。

25 (11) TRB暗号鍵リフレッシュ命令 : Refresh TRB Key command

命令形式：REFRESH\_TRB\_KEY

入出力レジスタ：なし

動作：T R Bを外部に記録する際にはかならず用いる対称暗号法の暗号鍵 Ktrb とページテーブルの署名確認に用いる暗号鍵 Ktlb を新たな乱数値に置き

5 換える。

この命令は、Ktlb や Ktrb が漏洩する可能性がある場合などに使用する。ただし、T R Bを用いてライセンス情報などを暗号化して管理していた場合、この命令後は以前の暗号化情報を復号できなくなるので、注意が必要。このような場合には例えば、リフレッシュの前に別の鍵で暗号化して退避しておくなど

10 の処理が必要になる。利用例の詳細は後述。

(1 2) 保護領域のキャッシュを優先する命令：

保護データの暗号・復号速度を重視する場合に利用する。

(1 3) プロセス間で漏洩しない熱乱数生成命令：

(1 4) 内蔵暗号・復号コマンド

15 G T 1 0を実現のためにC P U内部に実装した対称鍵暗号法や公開鍵暗号法の暗号化・復号命令をインストラクションセットとしてソフトウェア開発者に見せることにより、ハードウェアリソース利用効率が高まる。

(1 5) 定期割り込み間隔設定コマンド

20 G T内蔵水晶発信機が発生する内部割り込みの間隔を設定する。外部の信頼できる時計と安全に同期するセキュアクロックを耐攻撃プロセスとして実現する場合に用いることができる。

<例外処理>

G T 1 0を実現するC P Uは、従来の例外処理に加え、次のような例外処理を実装する。例外処理には、大きく分けて、耐攻撃ページアクセス権関連例外

25 と、耐攻撃領域復帰関連例外と、レジスタ封印関連例外と、ハッシュ関連例外

等とがある。以下、順に各例外処理について説明する。

・耐攻撃耐攻撃ページアクセス権関連例外

(1) 証明書指定誤り

この例外処理は指定した番号あるいは識別情報の GT 証明書 (Cgt) がない場  
5 合に発生する。

(2) GTライセンス認証フォールト

この例外処理はGTライセンスの認証に失敗した場合に発生する。

(3) 耐攻撃空間ページフォールト

TLB内の耐攻撃性フラグ tr がオフ（耐攻撃モードではない）または、その  
10 ページに対応する行がTRB内に無いのに耐攻撃領域へのアクセス命令を実行  
しようとした。

(4) TLB署名不一致例外

TLB.sign (TLB内の sign フィールド) の値が、ブロックに含まれる署名と  
一致しない。本例外が発生した場合、GT10は自動的に当該TLBのページ  
15 テーブルからのロードを停止する。さらに、TLB内の対応する行の有効性フ  
ラグ (TLB.p) 及び耐攻撃性フラグ (TLB.tr) をオフにする。

(5) アクセス権不正拡大例外

AUTHORIZE 命令で許可されたアクセス権よりも指定されたアクセス権の範囲が  
広い。本例外が発生した場合、GT10は自動的にアクセス権設定命令を拒否  
20 する。

(6) 暗号化キーテーブルアクセスフォールト

この例外は、TRB中に存在しない kid その他の検索キーを指定した場合、  
又はTRBのページアウトに失敗した場合に発生する。

(7) ライセンス削除フォールト

25 この例外は、ライセンスの削除に失敗した場合に発生する。

・耐攻撃領域復帰関連例外

(8) 耐攻撃コード復帰例外

:RETURN\_TO\_TAMPER\_RESISTANT\_CODE\_EXCEPTION

- call、jump 又は return によって、一般の仮想領域又は耐攻撃領域からコード復帰鍵 Kc が異なる耐攻撃領域に実行中アドレスが移動した際には、必ず耐攻撃コード復帰例外が発生する。この例外の処理において、移動先耐攻撃領域の例外アドレスである TRB.eadr (TRB内の eadr フィールド) の内容が確認され、そのフィールドに例外アドレスが存在する場合、プロセッサコアは、その例外アドレスにアクセスする。例外アドレスが存在しない場合には、当該 TRB行が削除され、耐攻撃コード復帰例外アドレス不正例外が発生する。

(9) 耐攻撃コード復帰例外アドレス不正例外

- この例外は、耐攻撃コード復帰例外アドレスで指定された空間が、耐攻撃空間に存在しない場合に発生する。保護プロセスへの復帰時等に本例外が発生した際には、そのアドレスに対応する TRB内の行の有効性フラグ(p)をオフにし、OSなどによって指定されたアドレスへ強制的にジャンプする。

・レジスタ封印関連例外

(10) レジスタ封印済例外

この例外は、指定されたレジスタが既に同じ Kc を持つコードにより封印済みである場合に発生する。

(11) レジスタ解放済例外

この例外は、指定レジスタが封印されていない場合に発生する。

(12) 封印レジスタアクセスフォールト

:SEALED\_REGISTER\_ACCESS\_FAULT\_EXCEPTION

- この例外は、指定されたレジスタが既に Kc が異なる他のコードによって封印済みである場合に発生する。

・ハッシュ関連例外

(13) ハッシュ非実装例外

この例外は、GT10がハッシュ生成機能やチェック機能を実装していない場合に発生する。

5 (14) ハッシュ不一致例外

この例外は、GT10が、保護コード又は保護データをキャッシュする際にハッシュ値の不一致が生じた場合に発生する。

<動作>

以下、GT10によって行われる処理について説明する。

10 <GTの認証>

まず、GT10の認証手順について説明する。なお、以下の説明において、PKIの内容の理解を前提としている。

図16に、GTの認証を説明する図を示す。図16に示すように、GTの認証には、保護アプリケーション50、DRM30及び40、暗号モジュール  
 15 (PKIライブラリ20に含まれる)の各ソフトウェアパッケージを作成する各開発メーカー、GT10を作成する開発メーカー、PKIライブラリモジュール用認証局CApki、DRM用認証局CAdrm、DRMを利用するアプリケーション向け認証局CAap及びGTを認証する認証局CAgtが介在する。図16において、細い矢印は、データの流れを示し、太い矢印は、ソフトウェアパッケージへのライセンス又は証明書の埋め込みを示す。また、図16中の括弧中の数字は処理の順番を示す。また、各TRMソフトウェアは、各TRMソフトウェア開発メーカーが秘密裏に生成した対称鍵暗号方式の暗号鍵で暗号化されているものとする。

20

GTライセンスの生成と利用は次のような各システム運用とプログラムソフトウェア実行許諾手順によって実現される。

25



(1) GTメーカーは製品化するGT10の個別公開鍵 KPgt をGT専用の認証局 CAgt に渡す。さらに、GTメーカーは、GT10内部には秘密鍵 Kgt を埋め込む。

(2) GT専用の認証局 CAgt は、GT10の公開鍵証明書 Cgt をGTメーカーに発行する。なお、GT10の公開鍵証明書 Cgt は、クラス秘密鍵 Kcgt で署名され、クラス公開鍵 KPcgt は、GT専用の認証局 CAgt のルート秘密鍵 Kagt で署名された証明書の形で公開されていることとしてもよい。また、公開鍵証明書 Cgt において個別鍵とクラス鍵の双方を用いるのではなく、いずれか一方を用いることとしてもよい。クラス鍵がない場合、公開鍵証明書 Cgt はルート秘密鍵 Kagt で直接に署名されることとしてもよい。

(3) GTメーカーは、公開鍵証明書 Cgt をコピーし、TRMソフトウェア開発メーカーに配布する。

(4) 各TRMソフトウェアの開発メーカーは、開発したソフトウェア実行コードを鍵 Kc を用いて暗号化することにより、保護コードファイルを作成する。

(5) 各TRMソフトウェアの開発メーカーは、公開証明書 Cgt から取り出したGT10の公開鍵 KPgt を用いて鍵 Kc を暗号化し、(移動不可能な) GTライセンス Lxx を生成する。GTライセンスの構造については後述する。

(6) 各TRMソフトウェアメーカーは、ソフトウェアが実行される前にOSのイニシエーターがGTライセンス Lxx をGT10に投入するように、GTライセンス Lxx をそのソフトウェアパッケージに埋め込む。

(7) 各ソフトウェアの実行直前にそのGTライセンスがGT10に投入される。GT10は、公開鍵のペアにあたる秘密鍵 Kgt を用いて、GTライセンスを復号する。つまり、GT10内で各ソフトウェア開発メーカーによるGT10の認証が実行される。これにより、GT10上でそのソフトウェアを実

行することが許諾される。G Tで保護されるソフトウェアをフリーソフトウェア等にして流通させたい場合には、G Tライセンスを生成するためにクラス公開鍵 K P c g t を用いる事としてもよい。また、G Tで保護されるソフトウェアを特定のG T 1 0以外で使Ⓐ用できないようにしたい場合は、G Tライセンスを生成するために個別公開鍵 K P g t を用いる事としてもよい。

(8) G T 1 0は、G Tライセンスから取り出した鍵 K c とアクセス権 (Access rights) をG T 1 0内のT R B及びT L Bに設定する。ユーザは、たとえカーネルモード (スーパーバイザレベル/リング0) でも、T R Bに設定された K c を参照することはできない。

10     このように、T R Mソフトウェアパッケージには、G T 1 0に実行許諾を与えるためのG Tライセンスが埋め込まれ、暗号化されたプログラムコードが実行される前にそのG Tライセンスは、G T 1 0に投入される。G T 1 0は、個別秘密鍵 K g t を用いてG Tライセンスを復号し、そのG Tライセンスからプログラムコードの復号鍵 K c を取り出し、内部のT L BにリンクしたT R Bに、  
15     保護コードごとにその復号鍵 K c を維持する。G T 1 0において、暗号化されたコードは復号鍵 K c を用いて復号されながら実行される。

P K Iライブラリモジュール用認証局 C A p k i、D R M用認証局 C A d r m とD R Mを利用するアプリケーション向け認証局 C A a p から構成されるT R Mソフトウェアパッケージ向け認証局は、各パッケージのソフトウェアプロセスが保護  
20     データを交換する際にデータ転送先パッケージを認証するために利用する証明書を発行する。また、これらのうちD R M用認証局はライセンス転送先D R Mの認証に用いられることとしてもよい。

ここで示した4種類の認証局はすべて異なる運用にすることでコンテンツ (情報) 流通ビジネスのリスクが軽減することが可能となるが、そうした運用  
25     は必須ではない。なお、P K Iライブラリ2 0、デコーダD R M3 0及びメデ

ィアDRM40を含めて1つのOSとして評価し、このOS製品全体に対してGTライセンスを生成し、また公開鍵証明書を発行することとしてもよい。

- また、GTライセンスを保護実行コードと同じファイルに挿入することとしてもよいが、パッケージ超流通の便を考慮すると、暗号化された実行コードと
- 5 別のファイルとして管理することを推奨する。

次に、GTライセンスの構造について説明する。

- ライセンス生成側はライセンス許諾先GT（耐攻撃容器）の個別公開鍵 KPgt の証明書 Cgt を取得し、これを用いてGTライセンスを生成する。証明書 Cgt は X.509 準拠の形式としてもよい。個別公開鍵 KPgt の証明書 Cgt はクラス鍵
- 10 KPcgt で、また、クラス鍵 KPcgt の証明書 Ccgt は認証局 CAgt のルート公開鍵 KPagt で署名チェックした後、利用される。ライセンス生成側が生成する GT ライセンスの形式は以下のとおりである。

- ```
LicenseID || ACgt.ar ||
15      E (KPgt, Ks ) || E ( Ks, Key || LicenseID || ACgt || Is ) ||
      CgtID || CgtIDackey || else
```

ここで、各フィールドの意味は次のとおりである。

- E (K, D) : データ D を鍵 K で暗号化した結果
- 20 A || B : データ連結。データ A とデータ B が連結していることを示す。
- Ks : セッション鍵（乱数）。対称鍵（共通鍵）暗号法の鍵。
- Key : コード復号鍵／データ暗号化・復号鍵。Kc/Kd。
- LicenseID : ライセンスシリアル番号。ライセンス生成側がライセンスごとにユニークな番号を生成する。上位にライセンス生成者の ID を入
- 25 れ、中位にコンテンツの ID を入れることとしてもよい。

ACgt : G T アクセス条件。G T 内で強制可能なコード／データの利用条件を示し、次のフィールドを持つ。

ebim : 暗号化ブロック識別方式。図 5 に示す TLB. ebim フィールドにコピーされる。各値の意味は T L B のフィールドとして説明済みである。

5      aef : 被許諾コード鍵暗号化フラグ (ackey encryption flag)。on(1) なら被許諾コード鍵 ackey が個別公開鍵 KPgt を用いて暗号化されていることを示し、off(0) なら ackey の値がそのまま ACgt. ackey (ACgt の ackey フィールド) に入っていることを示す。

10      ackey : 被許諾コード鍵。暗号鍵 Kc 又は Kd で復号されたオブジェクトにアクセスが可能なプロセスのコードが記録されている保護ページの暗号鍵 key または暗号鍵 key を個別公開鍵 KPgt で暗号化した値。個別公開鍵 KPgt で暗号鍵 key を暗号化する場合で、GT が複数の KPgt を持つ場合には、さらにどの KPgt を用いているかを識別できる情報を付加する。なお、そのプロセスへのアクセス権の種類は、  
15      ar で指定される。

eadr : 例外処理アドレス。このフィールドはオプションである。key が異なるページからこの G T ライセンスの内容が設定されたページに復帰する直前に発生する例外処理コードの先頭アドレスである。

ar : 基本アクセス権。次のような値をとる。詳細は図 1 7 に示す。

- 20      (a) PR : ackey で指定されたプロセスからの読み出しのみ可能  
          (b) X : ackey で指定されたプロセスからの読み出しと実行が可能  
          (c) PRW : ackey で指定されたプロセスからの読み出しと書き込みが可能  
          (d) PWX : ackey で指定されたプロセスからの読み書きと実行が可能  
 25      能

uo : 外部出力禁止フラグ。on (1) ならこのライセンスのキー情報を格納した T R B 行は暗号化キーテーブルに出力 (ページアウト) されない。このフラグのデフォルトはオフ (0) であり、外部に出力可能であることを示す。

5 cl: キャッシュロックフラグ。オプション機能である。このフラグがオンである場合、この G T ライセンスで保護を指定された耐攻撃領域のデータはキャッシュの外に出ない。但し、ar が書き込み権を含まない (PR 又は X である) 場合には、本フラグは無効となる。このフラグのデフォルト値は、オフであり、外部に出力可能であることを示す。

10 df : デバッグモードフラグ (Debug mode Flag)。df が on なら、ar の指定に応じた動作を示す。T L B についての説明参照。なお、G T ライセンス内の ACgt の df をオンにする事により、保護コードをデバッグモードで実行する事ができる。また、超流通など形態で保護コードファイルを販売する場合には、df をオフにして販売することとしてもよい。

15 CgtID : KPgt の X.509 証明書の識別子値。issuer と serialNumber を連結した値。

CgtIDackey : オプション。ackey を暗号化した KPgt の X.509 証明書の識別子値。issuer と serialNumber を連結した値。ACgt.aef が off の場合には省略。また、CgtID と同値の場合にも省略可能。

Is : その他の情報

図 1 7 に、G T ライセンスに設定可能なアクセス権と被許諾権限の表を示す。耐攻撃性フラグ耐攻撃モードの際、プロセス (コード実行状態) からみた保護  
25 データ又は保護コード領域のアクセス権は、図 1 7 の中から選択され、T L B

に設定される。なお、図 17 における「指定コード」とは、当該 T R B 行内の Key フィールド又は Ackey フィールドで指定された鍵で復号可能なコードをいう。

また、図 17 において、「実行のみ可能」な権限がないが、F R など、「実行のみ可能」な権限と「読み出しと実行が可能」な権限の両方を指定できる C P U もある。その場合の耐攻撃権限は、それぞれ、P X 及び P R X と表現する事ができる。同様に、「書き込みが可能」な権限についても P W X 及び P R W X の両方が指定できることとしても良い。

G T ライセンスは、移動不可を条件としているため、移動機能や C R L (Certificates Revocation List : 証明書失効リスト) 及び L R L (License Revocation List : ライセンス失効リスト) をもたない。C R L や L R L を制御する必要がないため、G T の C P U への実装が容易になる。DRM の C R L 制御及び L R L 制御は O S 又はアプリケーションにて行われる。これにより、高い拡張性を維持する事が可能となる。

また、上述のように、ソフトウェア T R M をフリーソフトウェア等にした場合、G T ライセンスの生成にはクラス公開鍵を用いることとしても良い。また、逆に、特定の G T のみにソフトウェアを利用させたい場合には、G T 個別鍵に対応する公開鍵を用いることとしても良い。

以下、図 18 を用いて、上述の G T 認証処理における (8) の手順について詳しく説明する。G T 認証処理において、G T 10 は、アクセス許諾命令 (AUTHORIZE 命令) を実行することにより、G T ライセンスに基づいて T L B 及び T R B に鍵 Key やアクセス権等を設定する。

そのために、まず、G T 10 は、G T の公開鍵のペアにあたる秘密鍵 Kgt を用いて G T ライセンスを復号する (S1)。続いて、G T 10 は、正常に G T ライセンスを復号できたか否か判定し (S2)、正常に復号できた場合 (S

2:正常)、GT10は、指定仮想領域のTLBを検索する(S3)。正常に復号できなかった場合(S2:異常)、GT10は、GTライセンス認証フォールトを発生し(S11)、S16に進む。S16において、GT10は、エラー内容をresultAdrに設定し、メインフローに復帰する。

5 S3においてTLBを検索した結果、指定仮想領域が得られた場合(S4:正常)、GT10はTRBに空きがあるか否か判定する(S5)。S3の検索の結果、指定仮想領域が得られなかった場合(S4:なし)、GT10は、メモリが割り当てられていない旨を示す未割り当て例外を発生し(S12)、S16に進む。

10 S5においてTRBの空きがあった場合(S5:あり)、GT10は、GTライセンスに基づいて、TRB内のuo、cl、kid、key、ackeyの各フィールドに値を設定する(S6)。S5において、TRBの空きが無かった場合(S5:なし)、GT10は、TRB内の一部の行を暗号化キーテーブルに出力(ページアウト)する(S13)。GT10は、ページアウトに成功した場合(S14:成功)S6に進み、ページアウトに失敗した場合(S14:失敗)、暗号化キーテーブルアクセスフォールトを発生させた後(S15)、S16に進む。

15 S6の後、GT10は、TLBに格納する署名signを算出し(S7)、ar、tr、df、kid、ebim及びsignをTLBに設定する(S8)。続いて、GT10は、設定結果の署名を生成し(S9)、設定結果とS9で生成した署名とをresultAdrに設定し(S10)、メインフローに復帰する。

#### <保護コードファイルの構造>

以下、保護コードファイルの構造について説明する。暗号化がほどこされた保護コードファイル(実行形式)は、保護コードブロックおよび平文コードブロックの繰り返し構造の先頭にヘッダを付加した構造を持つ。また、保護コード

25 ドファイルを超流通に利用可能なファイルとする場合、そのファイルのヘッダ

には、さらに次のような情報が入る必要がある。

- ・コンテンツID：保護コードの暗号を解くためのコンテンツ復号鍵を持つGTライセンスとのリンク情報として必要である。

- ・コンテンツ暗号方式：保護コードの暗号化方式を特定する値として必要である。この値は復号鍵の長さを含むこととしてもよい。

図19に、超流通ファイル形式として用いる場合の保護コード実行形式の一例を示す。図19において、SCDF（超流通コンテンツ形式：Super Content Distribution Format）ファイルに、保護コード実行形式の内容が、SCDFの要素として格納されている。

#### 10 <保護コードのロード及び実行並びに保護データの退避と維持>

以下、図20を用いて、保護コードのロードと実行手順、及び保護データの退避と維持について説明する。

上述のように、暗号化が施された保護コードファイルは、ヘッダ、保護コードブロック及び平文コードブロックで構成される。保護コードファイルは実行の際にRAM17にロードされ、GT10（CPU）内での予測に従いブロックごとに命令キャッシュ12にコピーされる。このうち、ヒットした命令のみがプロセッサコア11で解釈され、実行される。図20に示すように、コードブロックのうち、保護コードブロックは、復号キャッシュラインで復号されてから命令キャッシュ12に記録される。

図20では、命令キャッシュ12とRAM17の間に、暗号ブロックを復号して命令キャッシュ12に書き込む復号キャッシュライン及び、平文ブロックを命令キャッシュ12に書き込む平文キャッシュラインの2種のキャッシュラインが備えられている。このように複数のキャッシュラインを備えることにより、処理を高速化することが可能である。

25 また、図20に示したように、保護コードの実行によって保護データをRA



M17に出力する際には、あらかじめ仮想メモリとして設定された耐攻撃領域に出力する。耐攻撃領域には、保護データを暗号化および復号するための対称鍵暗号法の鍵 Kd が関連付けられ、その鍵 Kd は秘密裏にGT10内に維持されている。鍵 Kd は、コード復号鍵 Kc ごとに異なる乱数として割り当てられる。

- 5 保護データは一旦GT10（CPU）内部のデータキャッシュ13に記録されたのち、暗号化されてからRAM17に出力される。また、RAM17上の保護データは、GT10（CPU）内部で復号されてデータキャッシュ13に記録され、そのうちヒットしたものがプロセッサコア11で利用される。

図20では、データキャッシュ13とRAM17の間に、暗号ブロックを復  
 10 号して命令キャッシュ12に書き込む復号キャッシュライン、データキャッシュ13の内容を暗号化してRAM17内の耐攻撃領域に書き込む暗号キャッシュライン、及び、平文ブロックを命令キャッシュ12に書き込む平文キャッシュラインの3種のキャッシュラインが備えられている。この構成によっても処理を高速化することが可能である。

- 15 なお、暗号化された保護データをRAM17からストレージ18内に退避させることが可能である。つまり、耐攻撃領域はRAM17内のみならず、バス18を介して接続されたストレージ19内にまでも拡張することが可能である。

#### <ページアクセス制御>

- 以上のようにして、GT10は、TRMプログラムコードを実行する許諾を  
 20 得て、その保護コードファイルを構成するコードブロックをRAM17にロードすると、保護コードブロックを復号して実行する。以下、保護コードを実行する際の保護コードを記録するページ（命令ページ又はコードページ）へのアクセス制御について図21を用いて説明する。図21において、矢印は、データの流れを示し、括弧内の数字は、処理の順番を示し、以下の説明における手  
 25 順の番号に対応する。

保護コードを記録するページへのアクセス制御は次のように行われる。

(1) 命令MMU 1 4 が、予測命令ポインタ (ip: instruction pointer) に記憶されている仮想アドレスを読み出す。

(2) 命令MMU 1 4 は、仮想アドレスに対応する物理ページ番号 ppn と耐  
5 攻撃フラグ tr と耐攻撃モードのアクセス権 ar を T L B から読み取る。

(3) 耐攻撃フラグが on の場合、命令MMU 1 4 は、T R B からコード復号鍵 (Kc) を取り出す。

(4) 命令MMU 1 4 は、暗号エンジン 1 6 に Kc を設定する。

(5) 命令MMU 1 4 は、キャッシュすべき保護コードブロックのアドレス  
10 を命令キャッシュ 1 2 と R A M 1 7 に指定する。

(6) R A M 1 7 から保護コードブロックを暗号エンジン 1 6 に投入する。

(7) 暗号エンジン 1 6 で復号された保護コードブロックは、命令キャッシュ 1 2 に記録される。

(8) 命令ポインタが記憶する仮想アドレスが予測命令ポインタに一致した場合 (つまりキャッシュヒットした場合)、プロセッサコア 1 1 は ip 用レジスタから T R S S フラグを読み出し、次の手順に進む。一方、キャッシュヒットしなかった場合、上述の (2) の手順に戻る。  
15

(9) プロセッサコア 1 1 は、ip 用レジスタから仮想アドレスを読み出す。

(10) プロセッサコア 1 1 は、命令MMU 1 4 からアクセス権 ar を取り  
20 出し、アクセス権を確認する。

(11) プロセッサコア 1 1 は、仮想アドレスの内容が記録された命令キャッシュ 1 2 内から保護コードブロックを読み出して実行する。

なお、手順 (10) において、実行する命令を記録する命令ページが切り替わった時には、プロセッサコア 1 1 は、手順 (11) で保護コードブロックの  
25 読み出しを行う前に、以下を確認する。

(a) 次に実行する保護コードブロックが記録されているページの暗号鍵 (TRB. key) または被許諾コード鍵 (TRB. ackey) が、直前に実行したコードブロックが記録されているページの暗号鍵 (TRB. key) と一致すること

(b) 次に実行する保護コードブロックについての TLB. ar に記録されたアクセス条件と、次に実行するアクセスとが合致すること

(a) 及び (b) の少なくともいずれかが不一致の場合には、プロセッサコア 11 は命令の実行を停止し、アクセス権例外を発生させる。

次に、保護データブロックが記録されているページへのアクセス制御について説明する。保護データブロックへのアクセス制御は、GT ライセンスの内容が TRB 及び TLB に設定されることによって強制される。保護データブロックへのアクセスは TRB 及び TLB に維持されている被許諾コード鍵 ackey、暗号・復号鍵 Kd、アクセス権 ar に従って制御される。

上述の保護コードブロックの場合のアクセス制御手順と、実行中の保護プロセスによる保護データブロックへのアクセス制御手順は、同様である。また、上記の手順 (10) において、実行する命令を記録するページが切り替わったとき、及びアクセスされる保護データを記録するページが切り替わったときには、保護データへのアクセスを実行する前に、プロセッサコア 11 は、次の点を確認する。

(a) アクセスされる保護データを記録するページの暗号鍵 (TRB. key) または被許諾コード鍵 (TRB. ackey) が実行コードを記録するページの暗号鍵 (TRB. key) と一致すること

(b) アクセスされる保護データを記録するページのアクセス権 (TLB. ar) と、次に実行するアクセスとが合致すること

(a) 及び (b) の少なくともいずれかが不一致の場合には、プロセッサコア 11 は、保護データへのアクセスを中止し、アクセス権例外を発生させる。

以下、図 2 2 を用いてプロセッサコア 1 1 によって行われる上述の保護コード及び保護データへのアクセス制御について、より詳しく説明する。

まず、プロセッサコア 1 1 は、例外／割り込みイベントの発生を待つ（S 2 1）。例外／割り込みイベントが発生すると、プロセッサコア 1 1 は、例外・  
5 割り込み処理を行い（S 3 5）、S 2 1 に戻る。なお、例外・割り込み処理について詳しくは図 2 3 に示す。図 2 3 に示される各種例外処理については、上記<例外処理>において詳しく説明したため、説明を省略する。

例外／割り込みイベントが発生しない場合（S 2 1：なし）、プロセッサコア 1 1 は、命令ページ切り替えが発生したか否か判定する（S 2 2）。

10 命令ページ切り替えが発生した場合（S 2 2：あり）、プロセッサコア 1 1 は、命令ページへのアクセス権の有無を判断する処理を行う（S 2 3）。命令ページ切り替えが発生していない場合（S 2 2：なし）、S 2 8 に進む。

S 2 3 の判断処理において、プロセッサコア 1 1 は、次に実行する保護コードブロックが記録されているページの暗号鍵 (TRB. key) または被許諾コード鍵  
15 (TRB. ackey) が、直前に実行したコードブロックが記録されているページの暗号鍵 (TRB. key) と一致し、且つ、次に実行する保護コードブロックについての TLB. ar に記録されたアクセス条件によって次に実行するアクセスが許可されているか否か判定する。上記 2 つの条件を満たす場合、プロセッサコア 1 1 は、次に実行するコードには、切り替わった命令ページへのアクセス権があると判  
20 断し（S 2 4：あり）、S 2 5 に進む。そうでない場合（S 2 4：なし）、プロセッサコア 1 1 は、ページアクセスフォールト例外をキューイングし（S 2 7）、S 2 1 に戻る。

S 2 5 において、プロセッサコア 1 1 は、次に実行する保護コードブロックが記録されているページの暗号鍵 (TRB. key) が、直前に実行したコードブロッ  
25 クが記録されているページの暗号鍵 (TRB. key) と一致するか否か判定し、さら

に、プロセッサコア 11 は、そのページに対応する TRB 内の行の eadr フィールドに既に値が設定されているか否か判定する。TRB.key が一致せず、且つ eadr フィールドに既に値が設定されている場合（S 25：YES）、プロセッサコア 11 は、耐攻撃コード復帰例外をキューイングし（S 26）、S 21 に  
 5 戻る。TRB.key が一致するか、又は eadr フィールドに値が設定されていない場合（S 25：NO）、S 28に進む。

S 28において、プロセッサコア 11 は、命令ページから命令を読み出して、その命令を解析する。さらに、プロセッサコア 11 は、現在実行されているコードが、命令の中で指定されたレジスタへのアクセス権を有するか否か確認する（S 29）。そのコードが指定されたレジスタへのアクセス権を有する場合  
 10 （S 29：あり）、S 30に進む。そのコードが指定されたレジスタへのアクセス権を有しない場合（S 29：なし）、プロセッサコア 11 は、封印レジスタアクセスフォールト例外をキューイングし（S 34）、S 21に戻る。

S 30において、プロセッサコア 11 は、レジスタで示されたデータページ  
 15 が、前のデータページと切り替わっているか否か判定する。切り替わっていない場合（S 30：なし）、命令を実行し（S 33）、S 21にもどる。なお、命令実行処理について詳しくは図 24に示す。図 24に示される命令については、上記<インストラクションセット>において詳しく説明したため、説明を省略する。

20 データページが切り替わっていると判定した場合（S 30：あり）、プロセッサコア 11 は、そのデータページへのアクセス権の有無を判断する処理を行う（S 31）。S 31の処理は、S 23で説明した保護コードの場合と同様であるので説明を省略する。S 31の判断の結果、プロセッサコア 11 は、次に実行されるコードには、切り替わったデータページへのアクセス権があると判  
 25 断する場合（S 32：あり）、S 33に進む。そうでない場合（S 32：な

し)、S 2 7に進む。

#### <保護ソフトウェアの起動>

次に、図 2 5 を用いて、G T 1 0 上で保護されるソフトウェア、つまり保護コードファイルを起動する際における、O S カーネル 6 0 及び G T 1 0 の動作について説明する。図 2 5 において、細い矢印がデータのリンクを示し、太い矢印がデータの流れを示す。また、括弧内の数字は、処理の順番を示し、以下の説明における手順の番号に対応する。

保護コードファイルを起動する手順は、以下のとおりである。

(1) O S カーネル 6 0 は、T L B を設定することにより、保護コード及び保護データをロードする領域となる仮想メモリ領域及び物理メモリ領域を確保し、仮想メモリと物理メモリ領域とをリンクさせる。

(2) O S カーネル 6 0 からの AUTHORIZE 命令に基づいて G T 1 0 は、T L B にリンクする T R B を設定する。

(3) O S カーネル 6 0 からの CALL などのコードモジュール呼び出し命令に基づいて、G T 1 0 は、ip レジスタを設定する（呼び出し先インストラクションにヒットしない場合、G T 1 0 は該当するコードブロックを復号し、復号されたコードをキャッシュにコピーする）。

(4) G T 1 0 は、ip で指定されたアドレスの命令を実行する権利を TLB. ar (T L B 内の ar フィールド) の内容で確認する。

(5) 命令を実行する権利を確認できた場合、G T 1 0 は、ip で指定されたアドレスの命令（図では STR 命令）を読み出し、デコードし、実行する。

#### <保護データへのアクセス>

次に、図 2 6 を用いて、G T 保護コードを実行するプロセスから保護データにアクセスするメカニズムについて説明する。この例では、保護データ領域は保護コード実行前に確保され、保護コードファイルから初期値もロードされて

いることを前提としている。また、OS 6.0からの AUTHORIZE 命令をGT10  
が実行する事により、保護データ領域へのアクセス許諾とアクセス権設定は、  
既に行われているものとする。

以下、例として、命令“STR r0, [r1]”を実行する（レジスタ r 0 の値が示す  
5 内容をレジスタ r 1 の値が示すアドレスに書き込む命令）場合について、保護  
プロセスが保護データにアクセスする手順と、それに対応するGT10の動作  
について説明する。図26において、矢印や括弧内の数字の意味は、図25と  
同様である。

（1）保護コードは、メモリアクセス命令（STR やLDR など）を実行する。

10 （2）GT10は、アクセスするアドレスに対応するTLB内の行中のアク  
セス権情報 TLB.ar（TLB内の ar フィールド）を確認する。

（3）保護コードは、TLB.vpn（TLB内の vpn フィールド）に一致する仮  
想ページ番号を持つページのデータにアクセスする。（なお、データがキャッ  
シュにない場合、保護コードは、その仮想ページに対応する物理ページからデ  
15 ータを復号してキャッシュにコピーする。）

<保護プロセスからの保護モジュールの呼び出し>

次に、保護プロセスから保護モジュールを呼び出す手順について説明する。  
保護プロセスから他の保護コードモジュール（DLL（Dynamic Link  
Library）など）を起動する場合には、次の2ケースがある。

20 （a）呼び出される保護コードモジュールのコード復号鍵 Kc が、そのモジ  
ュールを呼び出す保護プロセス（以下、親プロセスという）のコード復号鍵と  
同じケース

（b）呼び出される保護コードモジュールのコード復号鍵 Kc が親プロセス  
のコード復号鍵と異なるケース

25 （b）の場合に、保護親プロセスから他の保護コードモジュールを呼び出す

際に行う手順は図 2 7 に示すようになる。図 2 7 においても、矢印や括弧内の数字の内容は、図 2 5 と同様である。なお、図 2 7 において、保護コードモジュールは D L L となっているが、これは一例に過ぎない。

(1) 親プロセスは、T L B の設定などにより、保護コードモジュールをロードするための仮想領域と、その仮想領域に対応する物理領域を確保する。

(2) 親プロセスは、アクセス許諾 (AUTHORIZE) 命令により T L B にリンクする T R B を生成してコード復号鍵 K<sub>c</sub> を設定し、T L B にアクセス条件を設定する。

(3) 親プロセスは、利用している秘密情報用レジスタを、その親プロセスの保護データ領域内のスタック領域に格納する。(必要に応じ、保護データキャッシュ内のスタックデータは暗号化されて外部メモリに記録される。)

(4) 親プロセスは、レジスタ解放 (RELEASE\_REG) 命令を実行する。

(5) CALL 命令などにより、親プロセスは、保護コードモジュールを呼び出す。

(6) 親プロセスは、レジスタ封印 (SEAL\_REG) 命令を実行する。

(7) 呼び出しが返った場合、親プロセスは、退避したスタックをもとのレジスタに復帰させる。

図 2 8 に、上記手順において O S カーネルが行う処理のフローチャートを示す。

図 2 7 の手順の (1) において親プロセスが、保護コードモジュールをロードするための仮想領域と、その仮想領域に対応する物理領域を確保することを要求すると、耐攻撃領域取得システムコールが呼び出される。この要求の際、親プロセスは、必要となる領域のサイズと、G T ライセンスを O S カーネル 6 0 に通知する。このシステムコールにおいて、まず、O S カーネル 6 0 は、アプリケーションが利用しているレジスタの一覧を入力パラメタとする非許諾領



域スタックストア (STMEA\_TO\_UA) 命令をプロセッサコア 1 1 に出力する (S 9 1)。この S 9 1 は、図 2 7 の手順 (3) に対応する。

続いて、OS カーネル 6 0 は、アプリケーションが利用しているレジスタの一覧を入力パラメタとするレジスタ解放 (RELEASE\_REG) 命令をプロセッサコア 1 1 に出力する (S 9 2)。これにより、指定されたレジスタが解放される。この S 9 3 は、図 2 7 の手順 (4) に対応する。

OS カーネル 6 0 は、OS が利用しているレジスタの一覧を入力パラメタとするレジスタ封印 (SEAL\_REG) 命令をプロセッサコア 1 1 に出力する (S 9 3)。S 9 4 において、これら命令のパラメタが正しい場合 (S 9 4 : 正常)、これらの命令はプロセッサコア 1 1 によって実行され、S 9 5 に進む。そうでない場合 (S 9 4 : 不正)、S 1 0 3 に進む。

S 9 5 において、OS カーネル 6 0 は、親プロセッサによって指定された領域サイズのエントリをページテーブルに設定する (S 9 5)。指定されたサイズの領域を設定するための耐攻撃領域のリソースがあった場合 (S 9 6 : あり)、S 9 7 に進む。耐攻撃領域のリソースが不足していた場合 (S 9 6 : 不足)、S 1 0 3 に進む。

S 9 7 において、OS カーネル 6 0 は、GT ライセンスと、設定したアドレス及びページ領域を入力レジスタに設定し、アクセス許諾 (AUTHORIZE) 命令をプロセッサコア 1 1 に出力する。この S 9 7 は、図 2 7 の手順 (2) に対応する。プロセッサコア 1 1 が、正常に命令を実行した場合 (S 9 9 : 正常)、S 1 0 0 に進む。正常に命令を実行できなかった場合 (S 9 9 : 例外発生)、S 1 0 3 に進む。

S 1 0 0 において、OS カーネル 6 0 はアクセス許諾 (AUTHORIZE) 命令の結果を返却データとして設定する。この後、図 2 7 の手順 (5) が行われ、保護コードモジュールが呼び出される。続いて、OS カーネル 6 0 は、レジスタ

解放 (RELEASE\_REG) 命令をプロセッサコア 1 1 に出力し、OS が使用しているレジスタを解放させる (S 1 0 1)。最後に、OS カーネル 6 0 は、被許諾領域スタックロード (LDMEA\_FROM\_UA) 命令をプロセッサコア 1 1 に出力し、スタックされていたデータをレジスタにロードさせ、通常状態に復帰する。この S 9 5 は、図 2 7 の手順 (7) に対応する。

なお、パラメタの不正、リソースの不足及び例外が発生した場合、S 1 0 3 において、OS カーネル 6 0 はエラー内容を返却データとして設定し、S 1 0 1 に進む。

#### <耐攻撃コードの復帰時の例外処理>

10 call、jump 又は return 等の命令によって、一般の仮想領域又は耐攻撃領域からコード復号鍵 Kc が異なる耐攻撃領域に実行中アドレスが移動した際には、耐攻撃コード復帰例外が発生する。この例外処理の中で、次の 2 つの処理を実施する必要がある。

- (a) レジスタ封印の確認と、レジスタを解放済みの場合の封印の再設定。
- 15 (b) 耐攻撃セグメントセレクトの値の確認と、必要に応じた再設定。

GT 1 0 は、保護コードのレジスタ封印の前に、解除・復帰例外の先頭アドレスを TRB に設定する。また、外部からの割り込みによってレジスタが解放されたまま実行を継続することによって、レジスタに記録した秘密情報が漏洩することがないように、保護コードは解除・復帰例外処理を含む必要がある。この解除・復帰例外処理において、GT 1 0 は、封印したはずのレジスタが封印されているかを再度確認し、レジスタが封印されていないならば、封印しなおすか、保護コードの実行を中止するかしなければならない。

図 2 9 に、耐攻撃コード復帰例外処理による封印レジスタ不正アクセス防止のためのシーケンス例を示す。図 2 9 において、保護コードが不正なコードによって一時中断された後に復帰する際に、例外が発生した場合を示す。図 2 9

において、保護コードを中断する前にレジスタ封印 (SEAL\_REG r1) 命令が実行されているが、不正コードの実行中にレジスタ解放 (RELEASE\_REG r1) 命令が実行されたため、復帰時に封印したはずのレジスタが封印されていない。

図 29 に示す耐攻撃コード復帰例外のシーケンスの最初の 3 行は以下のとおりである。

```
TST RSSL.rlss, rlssBit
BNE chk_seg
SEAL_REG r1
```

この 3 行は、保護コードが利用するレジスタ r 1 の封印状態を R S S L レジスタ内のレジスタ r 1 に対応するビット rlss の値を調べる事により確認し、r 1 が封印されていない場合、r 1 を封印しなおす処理を示す。

また、不正なコードが割り込み等によってプロセスを奪い、データやスタックの T R S S フラグをオフ (0) に設定する可能性もある。T R S S フラグがオフになっているにもかかわらず保護コードが継続されると、耐攻撃領域ではなく一般仮想領域に保護データを書き込んでしまうことになる。このような事態を防止するために、耐攻撃コード復帰例外処理において、ip (命令ポインタ、CPUによっては PC (プログラムカウンタ) ともいう) が示す仮想アドレスごとの耐攻撃データ/スタックセグメントセレクトの再設定を行う必要がある。

図 29 に示す耐攻撃コード復帰例外のシーケンスにおける 4 行目から 6 行目は以下のとおりである。

```
ch_seg CMP ip, trSegmentHead
BMI ret
ORR trdss, trdss, trBit
```

この 3 行が、ip が示す仮想アドレスごとの耐攻撃データ/スタックセグメントセレクトの再設定を行う処理を示す。耐攻撃コード復帰例外処理が終了する

と、プロセスは保護コードに復帰する。なお、図 29 のシーケンスの最後において、不正なコードによって再度プロセスが中断され、不正なコードが保護コードが利用するレジスタ `r1` の内容を一般仮想領域に書き込もうとしている (`MOV r1, [unprotectedArea]`)。しかし、耐攻撃コード復帰例外処理において、

- 5    レジスタ `r1` は封印しなおされているため、封印レジスタアクセスフォールト (`SEALED_REGISTER_ACCESS_FAULT_EXCEPTION`) が発生している。

このように、耐攻撃コード復帰例外のシーケンスによって、`GT10` は、復帰時に保護コードの秘密データを守っている。

<保護コンテキストの切り替えの管理>

- 10    以下、耐攻撃ユーザプロセスから OS カーネルプロセスへのコンテキスト切り替えがおこり、全レジスタがスタックに退避されるケースを想定する。図 30 に、OS カーネルによる保護コンテキスト切り替え時のシーケンスの一例を示す。

- 図 30 において、アプリケーション 1 からアプリケーション 2 に保護コンテキストを切り替えると仮定している。当初、アプリケーション 1 が利用するレジスタ `r1` は、封印されている。その後、保護コンテキストを切り替える際に、OS カーネル 60 は、非許諾領域スタックストア (`STMEA_TO_UA`) 命令を実行することにより、レジスタ `r1` の値はデータ暗号・復号鍵 `Kd` で暗号化されてから退避される。さらに、スタックされたレジスタ `r1` は、レジスタ解放
- 15    (`RELEASE_REG`) 命令により 0 クリアされる。レジスタ `r1` が解放されることにより、アプリケーション 2 のプロセスがレジスタ `r1` を利用することが可能になる。

- その後、保護コンテキストが元のプロセスにもどる際に、OS カーネル 60 は非許諾領域スタックロード (`LDMEA_FROM_UA`) 命令を実行することにより、
- 25    退避されていたレジスタの値が復号されて、レジスタ `r1` に戻される。その後、

レジスタ r1 は再度封印される。コンテキスト切り替えの際に、このような保護コンテキスト管理によって、封印レジスタを含めたレジスタの維持、復帰を保障することは OS カーネル 60 の処理として位置付けられる。

5 なお、OS カーネル 60 が非保護システムコールを呼び出す際は、SP を一般仮想空間のアドレスに切り替えてから呼び出すなどの工夫が必要である。OS は従来通り、システムコールの戻り値を一般仮想領域に記録すればよい。

一方、OS カーネル 60 が保護システムコールを呼び出す際には、戻り値を格納するスタック領域を共有するための GT ライセンスをパラメタなどとして渡す。OS は受け取った GT ライセンスをパラメタとして AUTHORIZE コマンド  
10 を実行することで、耐攻撃スタック領域をアプリケーションプロセスと共有することができる。

#### <保護データ領域の安全な共有>

上記において、耐攻撃空間内のプロセス同士であっても、ライセンスオーナーが異なるコード及びデータの領域間では、相互にアクセスすることができないと説明した。しかし、以下の耐攻撃領域共有機能を用いることで、GT 10  
15 で保護され、互いに異なるコード復号鍵 Kc を持つソフトウェアパッケージ、ライブラリ、オブジェクト及びプロセス等の保護コードモジュール間で、保護データを安全に交換できるようにすることができる。

図 3 1 に保護データ領域の共有の一例を示す。図 3 1 は、GT 10 上でプレーヤプロセス及びデコーダ DRM プロセスが動作している場合を説明する図である。  
20

デコーダ DRM プロセスによってデコードされたデータは、DRM プロセス用の仮想ページに書き込まれる。このデコードされたデータは、データ暗号鍵 Kd を用いて暗号化された後、RAM 17 内の共有物理ページに記録される。RAM 17 の共有物理ページに記録されたデータはデータ復号鍵 Kd を用いて復  
25

号されて、DRMプロセス用の仮想ページに書き込まれる。プレーヤ（再生アプリケーション）プロセスはそのデータを読み出して再生する。

以下、このような保護データの共有を設定する手順について詳しく説明する。

- まず、保護データを共有するために、共有されることになる保護データ領域
- 5    を生成した生成元モジュールとその保護データ領域を生成元モジュールと共有する共有先モジュールを想定する。

- 生成元モジュールと共有先モジュールとがパッケージとして異なっている（すなわち  $K_c$  が異なる）にもかかわらず、共有先モジュールが生成元モジュールの保護データ領域を読み出すことができるようにするためには、図 3 2 に
- 10    示すようにして、生成元モジュールは共有先モジュールを認証しなければならない。その認証が正常に完了できた上で、G T 1 0 内部において、生成元モジュールの保護データ領域の暗号鍵  $K_d$  を共有先モジュールが利用できるように T L B および T R B が設定されなければならない。

- 以下、図 3 2 に沿って、モジュールの認証、並びに、保護データ復号鍵共有
- 15    手順の設定手順について説明する。図 3 2 において、括弧内の数字は、処理の順番を示し、以下の説明における手順の番号に対応する。また、図 3 2 中の各記号の意味は次のとおりである。

- RN : 生成元モジュールが一時的に生成した乱数
- KPacm : ソフトウェアモジュール向け認証局のルート公開鍵
- 20    Kacm : ソフトウェアモジュール向け認証局のルート秘密鍵
- Kdp : 共有先モジュールの秘密鍵
- KPdp : 共有先モジュールの公開鍵
- C (KPdp, Kacm) : 共有先パッケージの公開鍵証明書
- Kc1 : 生成元モジュールのコード復号鍵
- 25    Kc2 : 共有先モジュールのコード復号鍵

(1) 生成元モジュールが共有先モジュールに一時的に生成した乱数を渡す。

(2) 共有先モジュールが、自身のプロセスが生成した仮想領域の ID と Kc2 を KPgt で暗号化した値と乱数とを連結したデータに、そのデータのデジタル署名と署名に用いた秘密鍵に対応する公開鍵の証明書とを付加する。この証明書は PKI ライブラリ用認証局 (CApki)、DRM 用認証局 (CAdrm) あるいはアプリケーション向け認証局 (CAap) などの信頼できる第三者が発行したものでなければならない。なお、この説明では、証明書には、復号鍵 Kc="YYYYYY" が KPgt で暗号化されているとする。

(3) 共有先モジュールは、(2) で生成したデータを、生成元モジュールに一般のデータ共有／プロセス間通信などを利用して渡す。

(4) 生成元モジュールは、署名をチェックし、その正当性を確認できた場合、受け付けた被許諾コード鍵 Authorized Code Key とアクセス権 'PR' (耐攻撃モードで読み出し専用) と暗号・復号鍵 Kd を埋め込んだ G T ライセンスと指定仮想領域をパラメタとして、アクセス許諾命令 (AUTHORIZE) を実行する。なお、この説明では、暗号・復号鍵 Kd="AAAAAA" が G T ライセンスに埋め込まれているとする。

(5) アクセス許諾命令を実行した結果、共有先モジュールの T L B にリンクした T R B 内の行に、G T ライセンスの内容が設定される。これにより、共有先モジュールが共有保護領域から読み出すデータは、鍵 Kd で正常に復号された状態でキャッシュされる。

この結果、図 3 2 に示す例によれば、T L B の 4 行目には、物理ページ番号 "002" へのアクセス権 "PR" が設定され、この T L B 内の行に対応する T R B の 4 行目には、データ鍵 Kd="AAAAAA" が Key フィールドに設定される。これにより、生成元モジュールの保護データ領域 (T L B の 2 行目に対応) を、共有先モジュールが、データ鍵 Kd="AAAAAA" を用いて復号して読み出す事が可能とな

る。

なお、異なるモジュール間で相互認証を行う場合には、手順（２）において共有先モジュールは、生成元モジュールの公開鍵または公開鍵で暗号化した対称鍵暗号法の鍵で、生成した送信メッセージを暗号化して、生成元モジュール  
5 に渡せばよい。これにより、このメッセージは生成元モジュール以外では復号できないこととなる。

なお、図１０に示すTLB内で、IDが２である行とIDが７である行は、保護領域を共有した場合を例として示したものである。

図３３に、耐攻撃領域共有システムコールを呼び出した際の処理の手順を示  
10 す。この手順（S１１０からS１２３）は、システムコール時の入力パラメタが領域サイズの代わりに領域の先頭のアドレスであることをのぞいて、図２８に示す手順（S９０からS１０３）と同様であるため、説明を省略する。

なお、コンテンツ再生(Player)アプリケーションは、上述のローカルデータ共有方式を用いてデコーダ DRM４０からデータを得て、これを再生し、さらに  
15 編集することとしてもよい。編集し、その結果を記録する場合には、GTライセンスにおいて指定されたアクセス条件に従わなければならない。また、再生アプリケーションはこれらの処理をGT保護コードとして生成しなければならない。

再生時間や期限の制限を再生アプリケーションに強制するためにはセキュア  
20 タイマーが必要である。セキュアタイマーの構築は、GT１０の外に独立したハードウェア TRM として実現することとしてもよい。または、GT１０が水晶発信機などを内蔵することが可能である場合、上述の定期割り込み間隔設定コマンドを用いて耐攻撃ソフトウェアとして構築することとしてもよい。いずれの場合も、タイマーがなんらかの理由で停止する場合を想定すると、外部の  
25 Trusted Secure Timer と同期をとる機能を持つ必要がある。



### <DRMソフトウェア構築モデル>

以下、DRMソフトウェアモジュールの構築モデルに説明する。デコーダDRM30、メディアDRM40及びこれらのDRMが用いるPKIライブラリ20内の暗号・復号ライブラリ、更には、TRM化が必要なその他のアプリケーションは、GT10上で保護されるGT保護コードとして流通し、実行される。これらのモジュールは、ほぼ全文を暗文とすることによりGT10で保護される。

図34に、DRMソフトウェアモジュールの構築モデルの一例を示す。このモデルでは、上記4つのモジュールが異なる開発メーカーで開発され、異なるコード暗号鍵 key (Kc)で暗号化されていることを想定している。図34において、細い黒矢印は、暗号化されたライセンスキーのやり取りを示し、細い白矢印は、復号されたライセンスキーのやり取りを示し、太い黒矢印は、暗号化されたコンテンツのやり取りを示し、太い白矢印は、複合されたコンテンツのやり取りを示す。括弧内の数字は、手順の順番を示す。以下、図34に沿って、コンテンツとそのライセンスキーのダウンロード、管理、再生手順について説明する。

(1) インターネット等のネットワークからダウンローダアプリケーションを介して、暗号化ライセンスキー及び暗号化コンテンツは記録媒体に記録される。

(2) ライセンスキーは暗号化された状態で、ダウンローダーを介してメディアDRM40に転送される。

(3) メディアDRM40内で復号されたライセンスキーは、後述の方法を用いて安全に管理され、暗号化された状態で記録媒体に記録される。ライセンスキーの再暗号化はGT10で保護されたPKI暗号ライブラリ20を用いて実施される。

(4) ユーザの再生要求があった場合、メディア DRM 4 0 は記録媒体からライセンスキーを取り出し、復号する。

(5) メディア DRM 4 0 はデコーダ DRM 3 0 を認証し、デコーダ DRM 3 0 と共有するセッションキーを用いてライセンスキーを暗号化してデコーダ DRM 3 0  
5 に転送する。なお、この鍵の共有及び手順 (7) の保護データの共有については<保護データ領域の安全な共有>において説明した。

(6) デコーダ DRM 3 0 は記録媒体から取り出した暗号化コンテンツと、メディア DRM 4 0 から得たライセンスキーとをパラメタとして PKI 暗号ライブラリ 2 0 に復号処理を依頼する。

10 (7) 復号されたコンテンツは共有保護データ領域を介して Player など再生アプリケーションに渡される。

(8) 再生アプリケーションがコンテンツを再生する。

以下、図 3 5 から図 4 0 を用いて、上述の手順をより詳しく説明する。

まず、図 3 5 及び図 3 6 を用いて、メディア DRM によって行われる処理に  
15 ついて説明する。

まず、メディア DRM 4 0 は、耐攻撃権限設定 (SET\_TR\_RIGHTS) 命令及びレジスタの封印 (SEAL\_REG) 命令を実行する (S 1 3 1 及び S 1 3 2)。続いて、メディア DRM 4 0 は、自身に埋め込まれた秘密情報を取り出す (S 1 3 3)。メディア DRM 4 0 は、取り出された秘密情報に対応する GT ライセンスが、ライセンスを記録するライセンス DB に格納されているか否か判定する  
20 (S 1 3 4)。既にライセンス DB に格納されている場合 (S 1 3 4 : あり)、S 1 3 6 に進み、まだライセンス DB に格納されていない場合 (S 1 3 4 : なし)、メディア DRM 4 0 は、その GT ライセンスをライセンス DB に設定した後 (S 1 3 5)、S 1 3 6 に進む。

25 S 1 3 6 において、メディア DRM 4 0 は、ライセンス管理用の耐攻撃デー

タ領域を生成する。S 1 3 6 の処理について、詳しくは、図 3 6 を用いて後述する。

ライセンス管理用の耐攻撃データ領域を生成できた場合（S 1 3 7 : NORMAL）、S 1 3 8 に進み、ライセンス管理用の耐攻撃データ領域を生成できなかった場合（S 1 3 7 : ERROR）、S 1 4 2 に進む。

S 1 3 8 において、メディアDRM4 0は、ライセンス管理用の耐攻撃データ領域を初期化し（S 1 3 8）、ライセンス受信要求、再生許諾要求又は終了要求を待つ。ライセンス受信要求を受けた場合（S 1 3 9 : あり）、メディアDRM4 0はライセンスを受信して（S 1 4 3）、S 1 3 9 にもどる。再生許諾要求を受けた場合、メディアDRM4 0は再生許諾を行い（S 1 4 4）、S 1 3 9 にもどる。終了要求を受けた場合（S 1 4 1 : あり）、レジスタ解放（RELEASE\_REG）命令を実行し（S 1 4 5）、処理を終了する。

ライセンス管理用の耐攻撃データ領域の生成に失敗した場合（S 1 3 7 : ERROR）、メディアDRM4 0は、エラー表示を出力装置（不図示）に出力し（S 1 4 2）、S 1 4 5 に進む。

次に、図 3 6 を用いて、図 3 5 の S 1 3 6 について説明する。ライセンス管理用耐攻撃データ領域を生成する際、まず、メディアDRM4 0は、コード暗号鍵 Kd を用いてアクセス権 PRW のGTライセンスを生成する（S 1 5 1）。続いて、メディアDRM4 0は、耐攻撃領域取得システムコールを呼び出す（S 1 5 2）。S 1 5 2 の処理については、上記のとおりである。

耐攻撃領域を取得する際にエラーが生じなかった場合（S 1 5 3 : なし）、途中に生成された署名を確認し（S 1 5 4）、署名が正しければ（S 1 5 5 : 一致）、正常にメインフローに戻る。耐攻撃領域を取得する際にエラーが生じた場合（S 1 5 3 : あり）、又は、署名が正しくない場合（S 1 5 5 : 不一致）、エラーをメインフローに返す。

次に、図 3 7 から図 3 9 を用いて、デコーダ DRM によって行われる処理について説明する。まず、デコーダ DRM 3 0 は、耐攻撃権限設定 (SET\_TR\_RIGHTS) 命令及びレジスタの封印 (SEAL\_REG) 命令を実行する (S 1 6 1 及び S 1 6 2)。続いて、メディア DRM 4 0 は、自身に埋め込まれた  
5 秘密情報を取り出す (S 1 6 3)。

さらに、デコーダ DRM 3 0 は、復号されたコンテンツ用の共有保護データ領域を生成する (S 1 6 4)。この S 1 6 4 について、詳しくは図 3 8 を用いて後述する。

S 1 6 4 の処理が正常に行われた場合 (S 1 6 5 : NORMAL)、デコーダ DR  
10 M 3 0 は、再生許諾要求、再生要求及び終了要求の何れかを受けるまで待つ。

S 1 6 4 の処理が正常に行われなかった場合 (S 1 6 5 : ERROR)、デコーダ DRM 3 0 は、出力装置 (不図示) にエラーが生じた旨を表示し (S 1 6 9)、S 1 7 5 に進む。

再生許諾要求を受けた場合 (S 1 6 6 : あり)、デコーダ DRM 3 0 は、メ  
15 ディア DRM 4 0 からコンテンツのライセンスキーを取得し (S 1 7 0)、S 1 6 6 に戻る。

再生要求を受けた場合 (S 1 6 7 : あり)、デコーダ DRM 3 0 は、コンテンツのライセンスキーを取得済みであるか否か判定する (S 1 7 1)。コンテンツキーを取得済みである場合 (S 1 7 1 : 取得済)、デコーダ DRM 3 0 は  
20 暗号化ブロックを復号し、そのブロックを共有する処理を行う (S 1 7 3)。

S 1 7 3 について詳しくは図 3 9 を用いて後述する。続いて、デコーダ DRM 3 0 は、再生アプリケーションに返却値を通知し (S 1 7 4)、S 1 6 6 に戻る。S 1 7 1 の判定において、まだライセンスキーを取得していない場合 (S 1 7 1 : なし)、出力装置 (不図示) にエラー表示を行い (S 1 7 2)、

25 S 1 7 5 に進む。

終了要求を受けた場合（S 1 6 8：あり）、S 1 7 5に進み、デコーダDRM30は、レジスタ解放（RELEASE\_REG）命令を実行し、処理を終了する。

次に、図38を用いて、図37のS 1 6 4について詳しく説明する。復号されたコンテンツ用の共有保護データ領域を生成するために、まず、デコーダDRM30は、デコーダDRM30によってデコードされたコンテンツを記録するための耐攻撃領域を取得するために耐攻撃領域取得システムコールを呼び出す（S 1 8 1）。この処理については既に説明した。耐攻撃領域が生成できた場合（S 1 8 2：NORMAL）、S 1 8 4に進む。そうでない場合（S 1 8 2：ERROR）、エラー表示を出力装置（不図示）に出力させ（S 1 8 3）、メインフローに復帰する。

S 1 8 4において、デコーダDRM30は、GT10に、暗号化されたコンテンツを復号するために用いるPKI暗号ライブラリを認証させる。なお、この認証手順は、＜保護データ領域の安全な共有＞において、図32を用いて説明した認証手順と同様である。

S 1 8 4の認証が正常に行われた場合（S 1 8 5：NORMAL）、S 1 8 7に進む。S 1 8 4の認証が正常に行われなかった場合（S 1 8 5：ERROR）、エラー表示を出力装置（不図示）に出力させ（S 1 8 6）、メインフローに復帰する。

S 1 8 7において、デコーダDRM30は、耐攻撃領域共有システムコールを呼び出す。S 1 8 4の処理が正常に行われた場合（S 1 8 8：NORMAL）、メインフローに復帰する。これにより、デコーダDRM30とPKIライブラリが、耐攻撃領域に書きこまれた保護データを共有することができるようになる。S 1 8 4の処理が正常に行われなかった場合（S 1 8 8：ERROR）、エラー表示を出力装置（不図示）に出力させ、メインフローに復帰する。

次に、図39を用いて、図37のS 1 7 3についてより詳しく説明する。ま

ず、デコーダDRM30は、再生アプリケーションは既に認証されているか否か判定する（S191）。既に認証されている場合（S191：認証済）、S196に進む。まだ認証されていない場合（S191：未認証）、S192からS195が行われる。この処理は、図38のS184からS188と同様であるため、説明を省略する。S192からS195の処理においてエラーが発生した場合（S193及びS195でERROR）、エラー表示を出力装置（不図示）に出力させ（S198）、メインフローに復帰する。S192からS195が正常に行われた場合、再生アプリケーションとデコーダDRM30とは耐攻撃領域を共有し、再生アプリケーションは、デコーダDRM30によって耐  
10 攻撃領域に書きこまれたコンテンツを読み出すことができるようになる。

S196において、デコーダDRM30は、暗号化されたコンテンツをPKI暗号ライブラリ20を用いて復号する。復号されたコンテンツは、共有化された耐攻撃領域に書き込まれる。復号が正常に行われた場合（S197：NORMAL）、メインフローに復帰する。復号が正常に行われなかった場合（S1  
15 97：ERROR）、S198に進む。

次に、図40を用いて、再生アプリケーションによって行われる処理について説明する。

まず、再生アプリケーションは、耐攻撃権限設定（SET\_TR\_RIGHTS）命令及びレジスタの封印（SEAL\_REG）命令を実行する（S201及びS202）。続いて、再生アプリケーションは、自身に埋め込まれた秘密情報を取り出す（S  
20 203）。続いて、再生アプリケーションは、デコーダDRM30に認証を要求する（S204）。この認証要求に基づいて、上記S192が行われる。

認証が正常に行われた場合（S205：NORMAL）、再生アプリケーションは、再生要求、その他の要求又は終了要求を受けるのを待つ。認証が正常に行われ  
25 なかった場合（S205：ERROR）、再生アプリケーションは、エラー表示を

出力装置（不図示）に出力させ（S 2 1 4）、S 2 1 6に進む。

再生要求を受けた場合（S 2 0 6：あり）、再生アプリケーションは、デコーダDRM30に暗号ブロックを復号するよう要求する（S 2 1 0）。デコーダDRM30からの返却値が正常な場合（S 2 1 1：NORMAL）、再生アプリケーションは、そのブロックを再生する（S 2 1 2）。コンテンツの再生が終了するまで（S 2 1 4：No）、S 2 1 0からS 2 1 2は繰り返される。コンテンツの再生が終了すると（S 2 1 4：Yes）、S 2 0 6にもどる。

再生要求以外の要求を受けると（S 2 0 7：あり）、再生アプリケーションはその要求を実行し（S 2 0 8）、S 2 0 6にもどる。終了要求を受けると（S 2 0 9：あり）、再生アプリケーションはGT 1 0にレジスタ解放（RELEASE\_REG）命令を実行させ（S 2 1 6）、処理を終了する。

#### <秘密情報の管理>

証明書が発行された公開鍵暗号法の公開鍵に対応する秘密鍵 Kdrm など、DRMコードパッケージ開発者が生成する秘密情報を維持・管理する方式を図4 1に示す。図4 1において、細い矢印はデータの流れを示し、太い白矢印は、鍵等の埋め込みを示す。また、括弧内の数字は処理の順番を示し、以下の説明における手順の番号に対応する。

特に公開鍵証明書に対応する秘密鍵は、CRLの指定時などに、特定のGTと特定のDRMのセットのみを停止する必要性から、本方式を用いる必要がある。本方式の手順はおおよそつぎのようになる。

（１）開発者が生成した秘密鍵（Kdrm など）は、クラス・個別いずれの場合も開発者が生成した対称鍵暗号法の鍵 Kprd で暗号化され、パッケージ内に埋め込まれる。

（２）GTライセンス化された、Kprd とアクセス条件' PR' とは、耐攻撃モードでのみ読み出し可能な形態で保護コード内に埋め込まれる。

(3) コード全体は、Kc で暗号化され DRM として保護パッケージ化される。

(4) Kc は KPgt で暗号化され、DRM ライセンスとして DRM パッケージに入れられる。

(5) DRM 実行時に VHTRM からこの GT ライセンスは GT 10 に投入され、GT 10 内部で Kprd が読み出される。

(6) Kprd で Kdrm は復号され、利用される。

これにより、開発者が指定した GT 上の指定プロセス以外のプロセスは、秘密鍵 Kdrm を読むことができなくなる。特定の GT が破られた場合には、破られた GT の公開鍵暗号法の公開鍵証明書のみを失効すればよい。従って、同じ  
10 DRM 向けのものでも、他の GT 向けの証明書 (Cdrm2) は正当に利用することができる。

本方式は DRM の秘密鍵のみでなく、他の保護パッケージの秘密鍵を管理する方法としても利用できる。

#### <UDAC ライセンス管理>

15 保護コンテンツと UDAC ライセンスに関する一般情報の管理方式は、UDAC-MB および UDAC-LA の方式を引き継ぐこととしてもよい。図 4 2 にライセンス管理のためのメモリアクセスの方式を示す。

図 4 2 に示すように、コンテンツ復号鍵などのライセンスの秘密情報は、RAM 17 内の保護データ領域に記録されて管理される。その保護データ領域内の情報は、ファイルとしてストレージやフラッシュメモリなどに記憶されることとしてもよい。秘密情報を記録する際にデータ暗号鍵 Kd で暗号化することにより、CPU (GT 10) の再起動時にもライセンスの秘密情報が安全に保護された状態で利用できる。このデータ暗号鍵 Kd は TRB 暗号鍵 Ktrb で暗号化された状態で、暗号化キーテーブルの行として一般の外部記憶装置に保持さ  
25 れることとしてもよい。



なお、Ktrb と Ktlb の耐攻撃能力を高めるためには、一定の期間を置いて Ktrb と Ktlb の内容をリフレッシュする必要がある。リフレッシュの前にはライセンス情報をすべて、耐攻撃プロセスが生成した一時キーで暗号化してバックアップしなければならない。リフレッシュによる Ktrb 及び Ktlb 変更のあと

5 に、ページテーブルと暗号化キーテーブルを再構築する。全ライセンス情報を復号し、再構築した耐攻撃領域にリストアすればよい。

上記の秘密鍵管理が前提として CRL の管理を行う事も可能である。また、ライセンスごとの CRL 制御機能は上記のライセンス管理機能内に持つこととしてもよい。1つの DRM パッケージの証明書は、GT 10 が持っている証明書の数だけ発行されることを原則とする。DRM パッケージ自体を失効する場合にはそれらの証明書のすべてが失効される。また特定の GT を失効する場合には、その GT 用に発行した DRM パッケージ証明書のみが失効される。

プログラムコンテンツを超流通し、UDAC ライセンスを扱う場合、DRM ソフトウェアを利用せず、簡易版超流通機能として、GT ライセンスをそのまま利用してもよい。また、GT 上に構築した DRM ソフトウェアを用いて、UDAC-MB や PI ライセンスとして処理してもよい。DRM ソフトウェアを用いる場合、UDAC-MB/PI ライセンスのうち、基本アクセス権(PR, X, PRW, PWX)のみは DRM 内で GT ライセンスに変換し、CPU のアクセス制御命令(AUTHORIZE)を用いて強制する。その他のアクセス条件については、GT 保護 DRM 内で強制処理を行う。

20 当然ながら、UDAC のみでなく、今日のソフトウェア TRM を用いた各社の DRM も GT 保護化することで、ハードウェア TRM レベルの耐攻撃性を持つことができる。

#### <変形例>

上記において、GT ライセンスは移動しない事として説明したが、DRM 間の

25 ライセンス移動機能の実現する事も可能である。同じ GT 上で実行される 2 つ

- のDRMプロセス間でのライセンス移動・許諾は、＜保護データの安全な共有＞で認証された共有耐攻撃領域を介して実現することとしてもよいし、UDAC-MB やUDAC-PI などのプロトコルを利用して実現する事としても良い。但し、ライセンスの移動機能を実装する場合、再送攻撃による不正コピーを防止するためには、TRM内に、以下の UPL (Unreplicatable Private Locker : 複製不能プライベートロッカー)にライセンス管理用秘密情報を格納しなければならない。この UPL を GT のみで実現したい場合には、TRB.uo (TRB内のuoフィールドの値) および TRB.c1 (TRB内のc1フィールドの値) を両方とも on にした耐攻撃領域にライセンス管理用秘密情報を格納することでこれが可能になる。
- 5      5      6      7      8      9      10      11      12      13      14      15      16      17      18      19      20      21      22      23      24      25
- ただし、GT の電源を切断後も UPL を用いて管理しているライセンスを維持する必要がある場合には、ライセンス管理用秘密情報を格納する UPL の領域だけでも不揮発性記憶素子にし、Permanent UPL (半永久 UPL) にする必要がある。耐攻撃空間の一部を Permanent UPL に割り当てる方法については、ここでは特定しない。
- 15      GT の外部に TRM 化した UPL を実現するためには、UPL はUDAC-MB などの TRM 認証プロトコルを UPL 内に実装する必要がある。外部に実現する Permanent UPL としてはUDACを実装したSecure MMCなどを利用することができる。
- 次に、第2実施形態について説明する。第1実施形態において、コードブロック及びデータブロックは、平文又は暗文であった (ebim=0又は1)。第2
- 20      実施形態によれば、ブロックは、暗文及び平文更には他の情報の組合せであってもよい。
- 以下、図43を用いて第2実施形態に係わるGTを実現するCPUの構成について説明する。
- なお、図43において、図3に示す第1実施形態と同様の部分は省略されている。図43に示すように、第2実施形態に係わるGT100は、第1実施形態
- 25      26      27      28      29      30      31      32      33      34      35      36      37      38      39      40      41      42      43      44      45      46      47      48      49      50

にかかわる構成に加えて、さらに、保護ブロックセクタ101及び104、ハッシュチェッカ102、及びハッシュエンジン103を備える。

保護ブロックセクタ101は、キャッシュ11又は12から出力されるコードブロック又はデータブロックが、保護されるべきブロックであるか否かを  
5 ebim の値に基づいて識別する。出力されたブロックが保護されるべきブロックである場合、保護ブロックセクタ101は、そのブロックをハッシュエンジン103に出力する。ハッシュエンジン103はそのブロックのハッシュを生成し、ハッシュが生成された後に、暗号ブロック16はそのブロックを暗号化し、RAM17に出力する。一方、キャッシュ11又は12から出力されたブ  
10 ロックが保護されるべきブロックではない場合、保護ブロックセクタ101は、そのブロックをRAM17に出力する。

また、保護ブロックセクタ104は、RAM17から出力されるコードブロック又はデータブロックが、暗号化されているブロックであるのか平文のブロックであるのか識別する。出力されたブロックが暗号化されているブロック、  
15 つまり保護ブロックである場合、保護ブロックセクタ104は、その保護ブロックを暗号エンジン16に出力する。暗号エンジン16は、その保護ブロックを復号し、復号されたブロックをハッシュエンジン103に出力する。ハッシュエンジンは、そのブロックのハッシュを生成し、ハッシュチェッカ102に出力する。ハッシュチェッカ102は、生成されたハッシュを確認し、ハッ  
20 シュが正しい事を確認できた場合、そのブロックをキャッシュ12又は13に出力する。一方、RAM17から出力されたブロックが保護ブロックではない場合、保護ブロックセクタ104は、そのブロックをキャッシュ12又は13に出力する。

なお、図43において、保護ブロックセクタ101及び104を備えること  
25 ととしたが、保護ブロックセクタ101及び104を1つの保護ブロックセ

レクタとして構成することとしてもよい。これにより、回路を小型化することが可能となる。

次に、第2実施形態に係わるTLB内のフィールドについて説明する。図5に示すTLBには、ebimフィールドが含まれる。第1実施形態によれば、この  
5 ebimの値は0又は1であった。第2実施形態によれば、このebimの取りうる値として、更に2から7が追加される。以下、ebimの値に応じるブロックの構造について説明する。

- a) ebim=0の場合、平文のみ（ダミーライセンス、第1実施形態と同様）
- b) ebim=1の場合、暗文のみ（第1実施形態と同様）
- 10 c) ebim=2の場合、保護ブロック開始識別コード使用
- d) ebim=3の場合、同上かつ暗号化ブロックのみ
- e) ebim=4の場合、署名付平文ブロックのみ。

キャッシュロード時署名確認必須

- f) ebim=5の場合、ハッシュ付暗号化保護ブロックのみ。  
15 復号後ハッシュ確認必須

- g) ebim=6の場合、保護ブロック開始識別コード使用。  
署名／ハッシュ確認必須

- h) ebim=7の場合、同上かつ暗号化ブロックのみ。ハッシュ確認必須  
すなわち、本フィールドの各ビットは、次のような意味を持つ。

- 20 ビット0：暗号化フラグ。このフラグがオンである場合、ブロックが暗号化されている事を示す。

- ビット1：保護ブロック開始識別コードフラグ。このフラグがオンである場合、ブロックに保護ブロック開始識別コードが付加されていることを示す。個  
フラグがオフである場合、GT10は、保護ブロック開始識別コードを認識す  
25 る必要はない。

ビット 2 : ハッシュ付加・確認フラグ。このフラグがオンである場合、データを GT 1 0 の外に掃出する際に、GT 1 0 は、そのデータブロックにハッシュを付加する。また、コードやデータを GT 1 0 内に取り込む際には、ブロックに付加されたハッシュ値を確認する。このフラグがオフである場合、ブロックにハッシュを付加したり確認したりする必要はない。

これらの ebim の値は、第 1 実施形態と同様に、GT ライセンスに含まれる被許諾コード鍵 ACgt 内の ebim フィールドの値に基づいて設定される。

次に、図 4 4 を用いて第 2 実施形態におけるブロックの構造について説明する。GT 1 0 によれば、保護コードブロックと保護データブロックの構造を同じ形式にしても良い。これにより、CPU 内の回路リソースの利用率を向上させることが可能となる。図 4 4 に示すように、開発者や創作者によって生成されたブロックは、乱数、一般命令やデータを含み、必要な場合さらにハッシュを含む。ebim が 1 又は 5 である場合、このブロックが暗号化されることにより、保護ブロックが生成される。ebim が 2、3、6 又は 7 である場合、このブロックが暗号化された後に、保護ブロックヘッド識別コードが平文で先頭に付加されることにより、保護ブロックが生成される。保護ブロックヘッド識別コードは、ブロックが保護ブロックであるか否かを示す暗号化フラグ及び、ブロックの最後にハッシュが付加されているか否かを示すハッシュフラグを含む。保護ブロックヘッド識別コード及び暗号化された乱数部分は、保護ブロック開始命令を構成する。

RAM 1 7 上の保護ブロックは、GT 1 0 内で復号されることにより、平文の一般命令やデータが取得される。ebim が 1 又は 5 である場合、乱数部分やハッシュ部分は、コードやデータのアドレスがかわらないように、NOP (No Operation) 命令に変換されてから、命令キャッシュ 1 2 又はデータキャッシュ 1 3 にロードされる。なお、ebim が 1 である場合は、第 1 実施形態と同じで

ある。ebim が 2、3、6 又は 7 である場合、乱数部分、ハッシュ部分に加えて保護ブロックヘッド識別コードも、NOP (No Operation) に変換されてから、命令キャッシュ 12 又はデータキャッシュ 13 にロードされる。

5     なお、プロセッサコア 11 上の作業データ等を暗号化した場合の保護ブロックの構造も、開発者又は創作者によって生成されたブロックを暗号化した場合の保護ブロックと同様の構造である。

図 4 5 に、ebim が 4 である場合のブロックの構造を示す。図 4 5 に示すように、ebim が 4 である場合、RAM 上のブロックは、平文のコード又は平文のデータに署名が付加された構造となっている。命令キャッシュ 12 又はデータ  
10    キャッシュ 13 上にブロックをロードする際には、署名は NOP 命令に変換される。なお、署名は、コード又はデータのハッシュ (SHA-1 など) を暗号鍵 Kc 又は Kd で暗号化した値とすることとしてもよい。なお、Kc 及び Kd は、GT ライセンスで指定され、TRB の暗号鍵フィールド (TRB. key) に設定されている。

15     次に、第 2 実施形態に係わる GT によって行われる処理について説明する。基本的な動作は、第 1 実施形態と第 2 実施形態とは同様であるため、以下の説明において、第 2 実施形態において更に付加された構成によって行われる処理に重点をおいて説明する。

まず、図 4 6 を用いて、保護ブロックセクタ 104 によって行われる処理  
20    について説明する。

まず、保護ブロックセクタ 104 は、RAM 17 (外部メモリ) からブロックのロード要求が出されるのを待つ (S 2 2 1 )。RAM 17 からブロックのロード要求が出されると (S 2 2 1 : 要求)、保護ブロックセクタ 104 は、予測されるブロックのアドレスと対応する行の ebim フィールドの値を  
25    TLB から読み込む (S 2 2 2)。

保護ブロックセクタ 104 は、ebim の第 1 ビットがオンであるか否か判定する (S 2 2 3)。ebim の第 1 ビットは、ブロックに保護ブロック開始命令が付加されているか否かをしめす。ebim の第 1 ビットがオンである場合 (S 2 2 3 : o n)、そのブロックには保護ブロック開始命令が付加されている。保護  
 5 ブロックセクタ 104 は、そのブロックの先頭に付加されている保護ブロック開始命令を読み出し (S 2 2 4)、その保護ブロック開始命令において暗号化フラグがオンであるか否か判定する (S 2 2 5)。暗号化フラグは、暗号化フラグがオンである場合 (S 2 0 5 : o n)、S 2 2 9 に進む。S 2 2 9 において、保護ブロックセクタ 104 は、そのブロックを暗号エンジン 16 に出力し、S 2 0 1 に戻る。この場合、そのブロックは、復号された後にキャッシュ 12 又は 13 に転送される。

暗号化フラグがオフである場合 (S 2 2 5 : o f f)、保護ブロックセクタ 104 は、さらに、ebim の第 2 ビットがオンであるか否か判定する (S 2 2 6)。ebim の第 2 ビットは、そのブロックを転送する際にハッシュの確認が必要であるか否かを示す。ebim の第 2 ビットがオンである場合 (S 2 2 6 : o n)、処理は S 2 0 9 に進む。ebim の第 2 ビットがオフである場合 (S 2 2 6 : o f f)、保護ブロックセクタ 104 はそのブロックをキャッシュ 12 又は 13 に転送する (S 2 2 7)。

S 2 2 3 の判定において、ebim の第 1 ビットがオフである場合 (S 2 2 3 : o f f)、保護ブロックセクタ 104 は、更に ebim の第 0 ビットがオンであるか否か判定する (S 2 2 8)。ebim の第 0 ビットは、そのブロックを暗号化することが必要であるか否かを示す。ebim の第 0 ビットがオンである場合 (S 2 2 8 : o n)、処理は S 2 2 9 に進む。ebim の第 0 ビットがオフである場合 (S 2 2 8 : o f f)、保護ブロックセクタ 104 は、さらに ebim の第 2  
 25 ビットがオンであるか否か判定する (S 2 3 0)。ebim の第 2 ビットがオンで

ある場合（S 2 3 0 : o n）、処理はS 2 2 9に進む。ebim の第2ビットがオフである場合（S 2 3 0 : o f f）、保護ブロックセクタ1 0 4は、そのブロックをキャッシュ1 2又は1 3に転送し（S 2 3 1）、S 2 2 1に戻る。

続いて、図4 7を用いて、ハッシュエンジン1 0 3によって行われる処理について説明する。

まず、ハッシュエンジン1 0 3は、ハッシュ要求のイベントが発生することを待つ（S 2 4 1）。イベントが発生すると、ハッシュエンジン1 0 3は、ハッシュの要求が行われたブロックを読み込む（S 2 4 2）。さらに、ハッシュエンジン1 0 3は、そのブロックのアドレスと対応する ebim を読み込み、その ebim の第2ビットがオンであるか否か判定する（S 2 4 3）。ebim の第2ビットがオンである場合（S 2 4 3 : o n）、ハッシュエンジン1 0 3は、そのブロックのハッシュを生成し（S 2 4 4）、そのブロックと生成したハッシュの内容を次の回路ブロックに転送し（S 2 4 5）、S 2 4 1に戻る。一方、ebim の第2ビットがオフである場合（S 2 4 3 : o f f）、ハッシュエンジン1 0 3は、ハッシュを生成しないでそのブロックを次の回路ブロックに転送し（S 2 4 6）、処理はS 2 4 1に戻る。

続いて、図4 8を用いて、ハッシュチェッカ1 0 2によって行われる処理について説明する。

まず、ハッシュチェッカ1 0 2は、ハッシュ要求のイベントが発生する事を待つ（S 2 5 1）。イベントが発生すると、ハッシュチェッカ1 0 2は、要求が行われたブロックを読み込む（S 2 5 2）。続いて、ハッシュチェッカ1 0 2は、そのブロックのアドレスと対応する ebim を読み込み、その ebim の第2ビットがオンであるか否か判定する（S 2 5 3）。ebim の第2ビットがオンである場合（S 2 5 3 : o n）、ハッシュチェッカ1 0 2は、ハッシュエンジン1 0 3によって生成されたハッシュと、そのブロックに付加されているハッシュ



ュとを比較する（S 2 5 4）。比較の結果、2つのハッシュが一致しない場合  
 （S 2 5 4：不一致）、プロセッサコア 1 1に、ハッシュ不一致例外が発生し  
 たことを通知し（S 2 5 5）、S 2 5 1に戻る。2つのハッシュが一致した場  
 合（S 2 5 4：一致）、ハッシュチェッカは、そのブロックを次の回路ブロッ  
 5 クに転送し（S 2 5 6）、S 2 3 1に戻る。

次に、第2実施形態の変形例について説明する。第2実施形態では、ebim に  
 基づいて、保護ブロックセクタは、ブロックを選択するとして説明した。こ  
 の方式以外の方式を以下に例示する。

1. 実行ファイルのヘッダに暗号化ブロックビットマップを埋め込む方法。
- 10 保護ブロックセクタは、このビットマップを先に読み込んで、ブロックを一  
 般ラインに出力するか、復号ラインに出力するかを判断する。
2. コードの先頭を平文にし、この先頭のコードにおいて、何ブロックの平  
 文のあと、何ブロックの保護コードがきてこれを繰り返すかを指定し、G T 1  
 0 0は、最初にこのコード（インストラクション）を実行する。このコードは、  
 15 途中で変更することも可能である。この方式によれば保護コードブロックセレ  
 クタの機能を縮小できる。アドレスの下位ビットだけで、保護コードをセレクト  
 することができれば、さらにセクタの機能を縮小させることが可能になる。

次に、第3実施形態について説明する。第1及び第2実施形態によれば、保  
 護コードや保護データがキャッシュ内に入る時には、乱数を変換した結果とし  
 20 て得られたNOPがキャッシュライン長ごとに存在することになる。これによ  
 り、キャッシュ内のリソースを無駄に使用するという問題が生じる。第3実施  
 形態は、その問題を解決する技術にかかわる。第3実施形態によれば、以下の  
 方法1から3によってキャッシュのリソースを有効に利用することを可能とな  
 る。

- 25 方法1) 乱数長とブロック数の積に、仮想領域長を加算した長さを、物理領

域長とする。(仮想領域長+乱数長×ブロック数=物理領域長)。RAM17上に仮想ページの大きさにはみ出す部分を持つことにより、はみ出した部分、つまり乱数を変換したNOPが、キャッシュ内に記録されないようにする。

方法2) 乱数長とブロック数の積に等しい長さを持つ領域を、NOP専用の

##### 5 物理領域として設定する。

- 方法3) キャッシュ内にNOPを記録しない。図49に、本方法によるNOPの処理を示す。図49に示すように、RAM17における、保護コードブロック又は保護データブロックは、暗号化されている。なお、場合によっては、保護コードブロック又は保護データブロックは、保護ブロックヘッド識別コードを含む。この保護コードブロック又は保護データブロックがGT10内で復号されると、一般命令(又は一般データ)、乱数及びハッシュを含むブロックが得られるが、キャッシュ内には、このうちの一般命令又は一般データが記録され、NOPが記録されるべき仮想アドレスのデータは記録されない。コードがNOPの仮想アドレスにアクセスした場合、NOPをコードに返す。なお、
- 10 仮想メモリに記憶されるNOPをOS等から読むことができるようにしても良いし、読むことができないようにしても良い。

次に、第4実施形態について説明する。第4実施形態は、上記において説明したGT10を2以上備えるCPU、つまりマルチCPUにおいて保護プロセスを動作させる場合に係わる。

- 20 マルチCPUで保護プロセスを動作させる場合、以下のような場合に備える必要がある。

1. コード復号鍵 Kc を1つだけ持つ保護プロセスが、複数のスレッドを複数のCPU上で平行に実行する場合

2. スヌープ機能による保護コードや保護データなど、キャッシュ内容の自
- 25 動同期に対応する。

図50に、GT10A及びGT10Bを備えるマルチCPUの構成をしめす。図50に示すように、GT10A、GT10B及びRAM17が、バスを介して接続されている。GT10A及びGT10Bは、それぞれキャッシュを備える。複数のスレッドをGT10A及びGT10Bによって並行に実行する場合、  
 5 実行に先立って、GT10A及びGT10Bは、DCTP (Digital Transmission Content Protection) の完全認証 (Full Authentication) 等の相互認証を行う事によってセッション鍵 K<sub>snp</sub> を得る。GT10A及びGT10Bは、秘密情報、保護コード及び保護データを交換する場合、セッション鍵 K<sub>snp</sub> を用いる。これにより、GT10A及びGT10Bは K<sub>trb</sub> や K<sub>c</sub>、K<sub>d</sub> 等を  
 10 安全に交換することが可能となる。GT10A及びGT10Bは、セッション鍵 K<sub>snp</sub> を用いてキャッシュ内のデータを暗号化し、互いに同期転送することにより、キャッシュ内容の同期を実現する。

次に、第4実施形態について説明する。従来のコンパイラやアセンブラを用いて作成したプログラムコードオブジェクトは、GT保護関連の情報を持たない。これを図51に示すような方式でGTでの保護コード実行形式、さらに  
 15 SCDF (超流通コンテンツ形式 : Super Content Distribution Format) に変換することができる。

図51に、コードオブジェクトから保護コード実行形式を出力するツール群の例を示す。図51では、保護コード実行形式とライセンスを生成し、さらに  
 20 実行形式を超流通実施のために SCDF 生成ツールを通して SCDF データを生成する例を提案している。

図51に示すように、SCDF生成ツールは、リンカ前処理部、リンカ、保護コード実行形式生成部、及びSCDF作成部を備える。まず、コードオブジェクトは、リンカ前処理部によって、複数のブロックに分割され、それぞれの  
 25 ブロックにNOP命令が追加される。続いて、リンカは、アドレス解決を行う。

さらに、リンカの後、保護コード実行形式生成部は、コード暗号鍵を用いて各ブロックを暗号化することにより保護コード実行形式を生成する。一方、GTライセンス生成部は、前記コード暗号鍵を含み、前記秘密鍵と対になる公開鍵によって暗号化されたライセンスを生成する。

- 5      次に、第5実施形態について説明する。GT10による保護を実現するためには、GT10と、GT10で保護されるDRMソフトウェアモジュールとが必要である。しかし当初はGTも普及しておらず、OSもGT10の機能をサポートしていないため、次のようなシナリオで普及を推進する必要がある。

（1）当初はソフトウェア TRM により、CPU 拡張部分のハードウェア命令エ  
10      ミュレータを開発し、従来のCPU向けのDRMソフトウェアをエミュレートする。これにより、従来のCPU向けのDRMソフトウェアも、GT10上で保護ソフトウェアとして稼動することが可能となる。

（2）DRM 部分は既存 DRM モジュールを流用する。

（3）OSがGT10の機能をサポートするまでは耐攻撃空間管理、保護コ  
15      ンテキスト切り替え管理やDRMなどをOSパッケージの外に持つ必要がある。

また、当初はアプリケーションとデコーダ DRM は一つのパッケージにし、Kc及びKdは同じとすることとしてもよい。

以下、上記各実施形態において説明したGTの応用例について、第6実施形態から第12実施形態として説明する。

20      GTの応用例について様々な例が挙げられるが、ここでは、例として、パーソナルコンピュータ、ワークステーション、PDA（Personal Digital Assistance）、携帯電話（簡易型携帯電話を含む）、スマートカード等のICカード、RFIDメディア、AV（Audio Visual）機器、GRID計算、ロボット等を例として挙げ、これらについて説明する。

25      まず、第6実施形態として、パーソナルコンピュータ及びワークステーショ

ンへのGTの応用について説明する。図52に、GT10又は100を、パーソナルコンピュータ又はワークステーションに実装した場合のシステム構成例を示す。図52に示すように、マザーボードにGT10又は100が搭載されている。システムコントローラには、USB (Universal Serial Bus) コントローラ、IDE (Integrated Drive Electronics) コントローラ、IEEE 1394 コントローラ、ビデオコントローラ及びオーディオコントローラ等が内蔵されている。

図52において、デジタルコンテンツを記録するメモ리카ード、ICカード、磁気ディスク、光磁気ディスク、デジタル多用途ディスク等の記録媒体には、メディアDRMチップが埋め込まれている。このメディアDRMチップは、GT10又は100上で保護されるモジュールである。このように構成することにより、特別に暗号・復号専用チップを組み込むことなく、GTをコンピュータに採用することにより、ハードウェアTRMと同じくらい強力に、デジタルコンテンツを保護することが可能となる。なお、図52において、システムコントローラとして North Bridge が記載され、インタフェースとしてUSBやIDEが記載され、シリアルバスとしてIEEE 1394が記載されているが、システム構成を限定する趣旨ではない。

さらに、ソフトウェア開発・追加のみで、個人認証、TCPA、電子財布、個人情報保護、Trusted GRID ComputingなどをハードウェアTRMと同程度に強力なセキュリティレベルで実現する。

また、GT保護ソフトウェアとして作成された電子投票ソフトウェアをPC等にロードする事により、PCから電子投票を行うことが可能となる。また、GT保護ソフトウェアとして作成されたファイル管理ソフトウェアをPC等にロードする事により、セキュアファイルシステムを構成することが可能となる。

次に、第7実施形態として、PDAや携帯電話等のモバイル機器への応用に

について説明する。

PC の TCPA (Trusted Computing Platform Alliance) 機能については、従来方式では別途特別なハードウェア装置を PC に接続する必要があったが、GT を PC に搭載すれば、その PC 上のソフトウェアで開発することのみで、この機能を

5 実現できる。

携帯電話の端末認証機能も、同様に従来方式より強力なセキュリティ強度で実現できる。例えば、携帯電話の SIM (Subscriber Identity Module) カードを交換する機能は、GT を実装する携帯電話間で保護データを交換するソフトウェアを用いる事により、より安全に実現される。

10 携帯電話や PDA (Personal Digital Assistance) などのモバイル製品に GT を適用する場合にも、特別に暗号・復号専用チップを組み込むことなく、ハードウェア TRM レベルの強力なコンテンツ保護を実現することができる。さらに、

もちろん、ソフトウェアを追加すれば、携帯電話や PDA に、個人認証機能、  
15 クレジットカード機能、プリペイドカード機能、個人情報保護、機能などを与える事ができ、これらの機能はハードウェア TRM と同等の強力なセキュリティレベルで実現される。

次に、第 8 実施形態として IC カードや RFID モジュールなどのセキュリティカードやモジュールへの応用について説明する。

20 IC カードについては従来方式では IC カード内のセキュリティ機能をカスタマイズするごとに TRM 化を施した個別のチップ生産を行う必要があった。しかも、個別のチップごとにハードウェア TRM の評価基準について審査を行う必要があった。

しかし、本実施形態によれば、GT に保護すべきセキュリティ機能を後から  
25 ファームウェアとして追加するだけで、ハードウェア TRM と同等のレベルの追

加セキュリティ強度を持つ I C カードを作成することができる。I C カードのセキュリティ評価についても、G T に追加したファームウェアについてのみ実施すればよい。

- I C カード、RFID モジュールなどのセキュリティカードやモジュールに G T
- 5 1 0 又は 1 0 0 を実装すれば、カスタマイズしたチップごとに TRM 化を施す必要はなく、CPU 部分のみ TRM 化を施しておけばよい。これで特別に暗号・復号専用チップなどを組み込むことなく、ハードウェア TRM レベルの強力なメディア DRM を実現できる。なお、ソフトウェア追加のみで個人認証機能、クレジットカード機能、プリペイドカード機能などをハードウェア TRM レベルの強力な
- 10 セキュリティで実現することも可能である。

さらに、CPU と比較して GT の寿命が短い場合に、GT のコア制御部分のみを交換できることとしてもよい。なお、GT コア制御とは、TRM、TLB 等にかかわる部分をいう。但し、この場合、CPU と I C カードを超高速バスで接続する必要がある。

- 15 次に、第 9 実施形態として A V 機器への応用について説明する。

- GT を A V 機器に搭載する場合、カスタマイズしたチップごとに TRM 化を施す必要はなく、CPU 部分のみ TRM 化を施しておけばよい。これで特別に暗号・復号専用チップを組み込むことなく、ハードウェア TRM と同程度の強度を持つ強力な DRM を実現することができる。また、さらに、A V 機器に G T に加
- 20 えて、ソフトウェアを追加することにより個人認証、オンライン課金機能などを A V 機器に与える事も可能である。これらの機能も、ハードウェア TRM と同程度の強度で実現することができる。

続いて、第 1 0 実施形態として、移動エージェント保護への応用について説明する。

- 25 GT は、エージェントが移動先で、不正に耐えて使命を全うするための保護

機能を実現することができる。つまり、GTは、耐攻撃エージェントとして、VHTRM に対して安全な移動機能を提供することができる。移動エージェントを耐攻撃化することにより、第11実施形態において説明する GRID 計算への GT 適用時と同様のセキュリティ機能を実現することができる。

5 続いて、第11実施形態としてGRID計算への応用について説明する。

GRID計算やネットワークエージェントでは、従来、以下の問題があった。

1. 完全性(Integrity) : GRID 計算の依頼先において、依頼した実行コードが正しい CPU で、正しいデータを用いて、正しく実行されているかどうかを確認することはできなかった。そのため、依頼先ユーザがどのような不正を働いたとしても、また依頼先の計算機に実行コードを書き換えるウィルスが入ったとしても、確実に確認する手段がなかった。

2. 秘匿(Confidentiality) : 計算の依頼先において、コードやデータの漏洩や不正コピーが発生する恐れがあった。

3. 課金(Accounting) : 計算の依頼先において、課金処理の不正や CPU 利用量データの改ざんが行われる恐れがあったため、課金処理や課金データの完全性を保証する必要があった。

4. 上記の問題を解決するためにソフトウェアTRMを用いると、処理が格段に遅くなってしまうため、処理速度を重視する GRID 計算の要件には見合わなくなっていた。しかも、計算機に関する特定の知識があればソフトウェアTRMを容易に破ることができた。

このような問題があるため、GRID 計算の依頼先として、一般のパーソナルコンピュータやワークステーションを積極的に利用することはできなかった。

しかし、GTを実装したCPU上で稼動する保護コードとして、計算依頼先で実行されるソフトウェアを開発することにより、以下が実現されるため、上記問題が解決される。



1. 安全なCPU（GT）の認証と実行コード改ざん防止保護
2. 計算処理改ざん防止機能
3. 安全な課金
4. 実行コードの不正コピー、不正利用の防止
- 5 5. 計算依頼先選択の最適化
6. 信頼の必要なものは TRM のレベルと証明書の発効日などを指定できるようにする。

第12実施形態として、ロボットへの応用について説明する。

- 人間や動物の作業や動作を肩代わりする自律型ロボットの研究やその発表が盛んであるが、その安全性についての検討もさらに重要になってくる。これまでは単なる情報処理装置としての計算機がウィルスなどにのっとられる脅威であったが、物理的に移動し、その用途によっては人間の力をも凌駕するロボットが同様の事態になることを想定する必要がある。しかし、ロボットに以下の機構を備える事により、このような問題を解決することが可能となる。

- 15 1. ロボット用の危険な部品の CPU にすべて GT を実装し、ロボット認定機関が発行した証明書が入っていることとする。
2. ロボット用の CPU は署名が確認できたコードしか実行しない機構を持つ。
3. ロボット用の CPU は証明書のない CPU とはセキュリティレベルの高い情報交換をしないこととする。
- 20 4. 「殺人・傷害禁止ルール」を GT 保護ソフトウェアとして実現し、認定機関が発行した証明書の秘密鍵を埋め込む。
5. 「殺人・傷害禁止ルール」実現ソフトウェアをルールに従って利用したアプリケーションのみに認定機関が発行した証明書の秘密鍵を埋め込む。
6. ロボットの全入力処理と危害を与えうる動作処理のプログラムコードすべてから、このルールエンジンを利用するようにし、全体を GT 保護化する。
- 25

7. プログラムコードにはかならずデジタル署名を付加して置くようにする。コードを実行する前に、署名チェックを行うようにし、実行許諾のない場合は停止する。

8. 統合ソフトウェア全体をGT保護化する。

- 5 これにより、ロボットを凶悪犯化するウィルスや中央制御機能改ざんなどに對抗することが可能となる。

次に、第13実施形態として、個人情報保護への応用について説明する。

- 今日、「.NET」のような万人からの信頼を獲得した個人情報管理サービスが多くの個人情報を管理しており、他のサービスは、自サービスを提供するため  
10 に必要な個人情報およびそれらの統計情報程度しか得ることができない。従って、これらのサービスは、営業戦略上の顧客統計情報を獲得したり、広告メールサービスを提供したりするためには、特定の個人情報管理サービスを利用する必要がある。このような状況におけるセキュリティ上の課題は次のようなものである。

- 15 1. 個人情報を回収するサービスが、サービス利用者に提示した「個人情報保護ポリシー」に従わない可能性がある。P3P (Platform for Privacy Preference : プライバシー制御のための基盤) を用いるなどにより、ポリシーを交換できたとしてもこれを強制する技術はない。

2. 個人情報を扱うサービスにおいて、個人情報を処理する従業員自身が不正な個人情報漏洩を行う可能性がある。  
20

- これら問題を解決するために、本実施形態によれば、各サービスを提供するサーバのCPUにGTを実装し、その上でサーバに、個人情報を扱うサーバDRMソフトウェアやサーバアプリケーションをGT保護コードとしてパッケージ化する。上述のように、GTライセンスには、ソフトウェアを実行するプロセス  
25 に対してアクセス条件を設定する事ができる。従って、これまで、アクセス制

御を外部から強制する事ができなかったサーバに対しても、アプリケーション操作におけるアクセス条件を強制する事ができるようになる。

- これにより、一般消費者に対して、信頼できる個人情報管理サービスを提供することができるようになる。さらに、サービスサイトへの個人情報の積極的な提供を促し、利用者の要求に従ったサイト間の積極的な個人情報交換を可能とする事により、広告・営業活動と消費活動をより活性化することが可能となる。

次に、第14実施形態として、ウィルス対抗手段への応用について説明する。

- 従来のウィルス対抗手段では、ソフトウェアのデジタル署名チェック機能と  
10 ウィルスチェックソフトウェアを併用している。しかしこの方式では、最新のウィルスが、署名チェック機能またはウィルスチェックソフトが起動するまでに動作する実行ファイルを書き換えることによる攻撃に対しては無力である。

- コード署名逐次確認機能を持つGTであれば、このようなウィルスに対抗できる。そのためには、CPUに第2実施形態に係わるGTを搭載し、実装コード  
15 ドチェックのためのソフトウェア（コードとデータ）をGTで保護する。そして、ブートからウィルスチェックソフトウェアが起動されるまでコード署名をGT（CPU）が逐次確認する。その上で、ウィルスチェックソフトウェアをGTの耐攻撃モードで実行する。ウィルスチェックソフトウェアは、各コードを実行する前にコードのハッシュを計算したり、署名を確認したりする。

- 20 これにより、新種ウィルスに対抗できなかったシステム上のセキュリティホールを壊滅することができる。なお、コード署名チェック機能については、既に説明した。

このように、GTをウィルス対抗手段に採用することにより、従来の方式に比較して格段に強度の高いウィルス対抗ソフトを実現することができる。

- 25 以上、GTの様々な応用例について説明した。このように、GTを搭載した

CPU を採用することにより、情報セキュリティ上の脅威により抑制されてきた各種先端的情報技術の社会基盤への適用が、促進されることが期待できる。例えば、次のような産業上の技術革新が期待できる。

1. コンテンツ保護の強度的及び機能的問題で積極的に PC に提供されなかったコンテンツが積極的にインターネットに流され、また各種 P2P (Person to Person) 技術を用いて積極的に超流通が促進される。

2. 利用者認証、電子財布などへの応用により、一般の情報機器のみを利用した買い物がソフトウェアの追加だけで安全にできる。これにより、インターネットでのお買い物を恐れていた利用者也買い物をするようになる。売る側も安心してサイトに品物を置いて販売できるようになり、市場がひろがり、グローバルなインターネット販売ビジネスを促進する。

3. 強力な個人情報保護機能の実現により、個人情報安心して積極的に売り込むことによるインターネットを介した広告活動と消費活動のさらなる活性化を期待できる。

4. 見ず知らずの CPU の利用を恐れたり、無償で CPU を貸し出す不公平感により積極的に計算パワーを提供できなかったりしていた GRID 計算を積極的に活用できるようになる。これにより、一般の共用 CPU の利用効率が 10 倍程度以上には向上すると考えられる。従って、単純に計算すれば、各コンピュータの処理速度が、ローカルな CPU だけで計算していた時に比較し、平均して 10 倍程度以上の速さが期待できる。あるいは、必要な計算に世界の CPU 利用を集中できるようになる。

5. ソフトウェア部品の「超流通」とネットワークを越えた自律協調型分散開発を目標とする「超流用 (Superappropriation)」を強力なウィルス対抗とモジュール利用料 P2P 課金のセキュリティで促進する。

6. GT による安全性の保障が、強力な実用ロボットの社会への浸透を積極的

に促進させる。

さらに将来、GT はいつでもどこでもだれにでも利用できる安全な汎用計算処理基盤として、高信頼ユービキタスコンピューティング (Trusted ubiquitous computing) のインフラにもなることができる。

- 5 以上、本発明の実施形態について説明したが、本発明は上述した実施形態に限定されるものではなく、他の様々な変更が可能である。

## 請求の範囲

1. プログラムを実行する中央演算装置であって、  
ブロックの暗号化及び暗号化ブロックの復号を行う暗号手段を備え、
- 5 前記中央演算装置には、第 1 の秘密鍵が秘密裏に隠され、  
前記暗号手段は、前記第 1 の秘密鍵と対になった公開鍵を用いて暗号化された第 1 のプログラムの第 1 のライセンスを、前記第 1 の秘密鍵を用いて復号することにより、前記第 1 のライセンスから前記第 1 のプログラムを構成する暗号化ブロックを復号するためのコード復号鍵を取得する、
- 10 ことを特徴とする中央演算装置。
2. キャッシュを更に備え、  
前記暗号手段は、前記第 1 のプログラムを構成する暗号化ブロックがメモリ領域から前記キャッシュに出力される際に、キャッシュ単位で前記暗号化ブロックを復号する、
- 15 ことを特徴とする請求項 1 に記載の中央演算装置。
3. ユーザが参照したり改ざんしたりすることができない耐攻撃バッファを更に備え、  
前記コード復号鍵は、前記耐攻撃バッファに記録される、  
ことを特徴とする請求項 1 又は 2 に記載の中央演算装置。
- 20 4. 前記第 1 のライセンスは、前記第 1 のプログラムの実行プロセスがメモリ領域にアクセスする際のアクセス条件を含み、  
前記中央演算装置は、前記第 1 のプログラムを構成する暗号化ブロックが記録される前記メモリ領域のアドレスと前記メモリ領域へのアクセス条件とを記録する TLB (Translation Look aside Buffer) と、
- 25 メモリ管理手段と、

キャッシュと、

プロセッサコアを更に備え、

前記TLBと前記耐攻撃バッファはリンクされ、

- 前記メモリ管理手段は、暗号化ブロックを記録するメモリ領域のアドレスに
- 5 基づいて前記TLBから前記メモリ領域へのアクセス条件を取得し、さらに、前記耐攻撃バッファから、前記メモリ領域に対応する前記コード復号鍵を取得し、

- 前記プロセッサコアは、前記メモリ管理手段が取得した前記アクセス条件に基づいて、前記実行プロセスから前記メモリ領域へのアクセスを実行することが許可されているか否か判定し、前記メモリ領域へのアクセスを実行する事が
- 10 許可されていると判定した場合、前記実行プロセスから前記メモリ領域へのアクセスを実行し、

- 前記暗号手段は、前記メモリ管理手段が取得した前記コード復号鍵を用いて前記メモリ領域内の前記暗号化ブロックを復号して得られたコードを前記キャ
- 15 ャッシュに書き込む、

ことを特徴とする請求項3に記載の中央演算装置。

5. 前記コード復号鍵と、前記暗号化ブロックを暗号化する際に用いられた暗号鍵は同一の鍵である、

ことを特徴とする請求項4に記載の中央演算装置。

- 20 6. 前記第1のプログラムの実行プロセスからアクセスされるメモリ領域が、第1のメモリ領域から第2のメモリ領域に切り替わった場合、

- 前記メモリ管理手段は、さらに、前記耐攻撃バッファから取得された前記第1のメモリ領域に対応するコード復号鍵と、前記第2のメモリ領域に対応するコード復号鍵とが一致するか否か判定し、一致すると判定した場合、前記実行
- 25 プロセスから前記第2のメモリ領域へのアクセスを実行し、一致しないと判定

した場合、前記実行プロセスから前記第2のメモリ領域へのアクセスを実行しない、

ことを特徴とする請求項4又は5に記載の中央演算装置。

7. 前記第1のライセンスは、前記第1のプログラムに埋め込まれている、

5 ことを特徴とする請求項1乃至6のいずれかに記載の中央演算装置。

8. 前記耐攻撃バッファには前記コード復号鍵ごとに、異なるデータ暗号鍵が記録されており、

前記暗号手段は、前記キャッシュ内のデータを前記メモリ領域に記録する際には、前記データを前記データ暗号鍵を用いて暗号化した後、前記TLBによって前記データ暗号鍵に対応付けられた前記メモリ領域に記録し、前記メモリ領域内の暗号化されたデータを読み出す際には、読み出された前記データを前記データ暗号鍵を用いて復号した後、前記キャッシュに書き込む、

10

ことを特徴とする請求項4乃至7のいずれかに記載の中央演算装置。

9. 第1のコードを実行することにより得られたデータを第2のコードで利用する場合、前記プロセッサコアは、前記データを記録するメモリ領域へのアクセス権を前記第2のコードに与えるように前記TLBを設定し、且つ、前記データを暗号化するためのデータ暗号鍵を、前記第2のコードが前記データを前記メモリ領域から読み出す際に使用するよう前記TLB及び前記耐攻撃バッファを設定する、

15

20 ことを特徴とする請求項8に記載の中央演算装置。

10. レジスタと、

レジスタへのアクセス制御を行うためのレジスタアクセス制御テーブルと、  
を更に備え、

前記プロセッサコアは、前記レジスタアクセス制御テーブル内の封印フラグ  
を用いて、前記レジスタの封印及び解放を制御する、

25



ことを特徴とする請求項 4 乃至 9 のいずれかに記載の中央演算装置。

1 1. T L B の内容を外部記憶装置内のページテーブルに記録する際には、  
前記暗号手段は、記録する内容に署名を付与し、

前記ページテーブルの内容を前記 T L B に取り込む前には、前記暗号手段は、

5 前記署名が正しいことを確認する、

ことを特徴とする請求項 4 乃至 1 0 のいずれかに記載の中央演算装置。

1 2. 前記耐攻撃バッファの内容を外部記憶装置内の暗号化キーテーブルに  
記録する際には、前記暗号手段は、記録する内容を暗号化する、

ことを特徴とする請求項 3 乃至 1 1 のいずれかに記載の中央演算装置。

10 1 3. 前記中央演算装置は、他の中央演算装置と接続され、

前記中央演算装置は、前記他の中央演算装置と相互に認証を行う事にことによりセッション鍵を取得し、前記中央演算装置の前記暗号手段は、前記セッション鍵を用いて前記中央演算装置の前記キャッシュの内容を暗号化して、前記他の中央演算装置に同期転送する、

15 ことを特徴とする請求項 1 乃至 1 2 のいずれかに記載の中央演算装置。

1 4. 前記暗号手段は、前記第 1 のプログラムを実行する前に、第 2 のプログラムに付加された前記第 2 のライセンスを前記公開鍵を用いて復号する事により第 2 の秘密鍵を暗号化する際に用いられた秘密鍵暗号鍵を取得し、さらに、前記取得された秘密鍵暗号鍵を用いて前記第 2 の秘密鍵を復号する、

20 ことを特徴とする請求項 1 乃至請求項 1 3 のいずれかに記載の中央演算装置。

1 5. 前記第 2 のライセンスには、前記第 1 のプログラムの実行プロセスから読み出しのみ可であることを示すアクセス条件が付加されており、

前記第 2 の秘密鍵は、前記第 1 のプログラムの実行プロセスからの読み出しのみ可能である、

25 ことを特徴とする請求項 1 4 に記載の中央演算装置。

16. 前記第2の秘密鍵は、データ暗号鍵によって暗号化されてメモリ領域に記録される、

ことを特徴とする請求項14又は15に記載の中央演算装置。

17. 前記耐攻撃バッファは、さらに、

5 対応する前記耐攻撃バッファ内の情報を前記耐攻撃バッファ外に出力しても良いか否かを示す外部出力禁止情報と、

対応する情報をキャッシュ外に出力しても良いか否かを示すキャッシュロック情報とを記録し、

前記外部出力禁止情報及び前記キャッシュロック情報に基づいて、第1のプログラム及び他のプログラムとの間における前記第1のライセンスの移動は管理される、

ことを特徴とする請求項3乃至16のいずれかに記載の中央演算装置。

18. 前記第1のプログラムは、トラステッドコンピューティングモジュールである、

15 ことを特徴とする請求項1乃至17のいずれかに記載の中央演算装置。

19. 前記第1のプログラムは、前記中央演算装置に電子財布を実現させるプログラムである、

ことを特徴とする請求項1乃至17のいずれかに記載の中央演算装置。

20. 前記第1のプログラムは、個人情報扱うプログラムである、

20 ことを特徴とする請求項1乃至17のいずれかに記載の中央演算装置。

21. 前記第1のプログラムは、前記中央演算装置の実装コードのウィルスチェックプログラムである、

ことを特徴とする請求項1乃至17のいずれかに記載の中央演算装置。

22. 前記第1のプログラムは複数の中央演算装置間を移動する移動エージェントである、

25

ことを特徴とする請求項 1 乃至 17 のいずれかに記載の中央演算装置。

23. 前記第 1 のプログラムを構成するブロックは、前記ブロックのハッシュ値の確認が必要であるか否かを示すハッシュ確認要否情報を含み、

前記ハッシュ確認要否情報に基づいて、前記ブロックのハッシュ値を算出し、

5 前記ブロックに付加するハッシュ手段と、

前記ハッシュ確認要否情報に基づいて、前記ブロックの前記ハッシュ値を確認するハッシュ確認手段と、

を更に備えることを特徴とする請求項 1 乃至 22 のいずれかに記載の中央演算装置。

10 24. 前記第 1 のプログラムを構成するブロックは、前記ブロックが保護を必要とするか否かを示す暗号化要否情報を含み、

前記暗号化要否情報に基づいて、前記ブロックを前記暗号手段に出力するか、そのままキャッシュ又はメモリ領域に出力するか判定する保護ブロック選択手段を、

15 更に備えることを特徴とする請求項 1 乃至 23 のいずれかに記載の中央演算装置。

25. 前記第 1 のプログラムの実行ファイルのヘッダは、前記第 1 のプログラムを構成するブロックの構成を示す暗号化ブロックビットマップを含み、

前記暗号化ブロックビットマップに基づいて、前記ブロックを前記暗号手段  
20 に出力するか、そのままキャッシュ又はメモリ領域に出力するか判定する保護ブロック選択手段を、

更に含むことを特徴とする請求項 1 乃至 23 のいずれかに記載の中央演算装置。

26. 前記第 1 のプログラムのコードの先頭は、前記第 1 のプログラムを構成する複数のブロックが、平文ブロックと暗号化ブロックの組合せの繰り返し  
25

であることを指定し、且つ、前記組合せにおいて平文ブロックが連続する数及び暗号化ブロックが連続する数を指定するコードであり、

前記コードを実行することにより、前記プロセッサコアは、前記ブロックを前記暗号手段に出力するか、そのままキャッシュ又はメモリ領域に出力するか

5 判定する、

更に含むことを特徴とする請求項 1 乃至 2 3 のいずれかに記載の中央演算装置。

2 7. 前記キャッシュとメモリの間に、

前記暗号手段を介するキャッシュラインと、

10 前記暗号手段を介さないキャッシュラインと、

を更に備えることを特徴とする請求項 2 乃至 2 6 のいずれかに記載の中央演算装置。

2 8. 請求項 1 乃至 2 7 のいずれかに記載の中央演算装置を有することを特徴とするコンピュータ。

15 2 9. 請求項 1 乃至 2 7 のいずれかに記載の中央演算装置を有することを特徴とする IC カード。

3 0. 前記第 1 のプログラムは、前記 IC カードのセキュリティ機能を実現するプログラムである、

ことを特徴とする請求項 2 9 に記載の IC カード。

20 3 1. 前記中央演算装置は、ロボットに搭載され、

前記第 1 のプログラムは、前記ロボットを制御する制御プログラムである、

ことを特徴とする請求項 1 乃至 2 7 のいずれかに記載の中央演算装置。

3 2. 中央演算装置に保護プログラムを実行する許諾を与える制御を行わせるプログラムであって、

25 前記保護プログラムは、コード暗号鍵によって暗号化され、

前記保護プログラムに対応して、前記コード暗号鍵を含み、且つ、前記中央演算装置に秘密裏に備えられた秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、

- 5 前記中央演算装置が前記保護プログラムを実行する前に、前記ライセンスを前記中央演算装置に投入し、

前記中央演算装置に備えられた暗号手段に、前記秘密鍵を用いて前記ライセンスを復号することにより、前記ライセンスから前記コード暗号鍵を取得させ、  
前記暗号手段に、前記コード暗号鍵を用いて前記保護プログラムを復号させる、

- 10 ことを含む処理を前記中央演算処理装置に実行させるプログラム。

3 3. 中央演算装置に、保護プログラムを実行する許諾を与える制御を行わせるプログラムを記録する記録装置であって、

前記保護プログラムは、コード暗号鍵によって暗号化され、

- 15 前記保護プログラムに対応して、前記コード暗号鍵を含み、且つ、前記中央演算装置に秘密裏に備えられた秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、

前記中央演算装置が前記保護プログラムを実行する前に、前記ライセンスを前記中央演算装置に投入し、

- 20 前記中央演算装置に備えら得た暗号手段に、前記秘密鍵を用いて前記ライセンスを復号することにより、前記ライセンスから前記コード暗号鍵を取得させ、  
前記暗号手段に、前記コード暗号鍵を用いて前記保護プログラムを復号させる、

ことを含む処理を前記中央演算装置に実行させるプログラムを記録する記録装置。

- 25 3 4. 中央演算装置に、保護プログラムを実行する許諾を与えるプログラム

実行許諾方法であって、

前記保護プログラムは、コード暗号鍵によって暗号化され、

前記保護プログラムに対応して、前記コード暗号鍵を含み、且つ、前記中央演算装置に備えられた秘密鍵と対になる公開鍵によって暗号化されたライセンス

5    スが存在し、

前記中央演算装置は、前記保護プログラムを実行する前に、前記ライセンスを取得し、

前記中央演算装置は、前記秘密鍵を用いて前記ライセンスを復号することにより、前記ライセンスから前記コード暗号鍵を取得し、

10    前記中央演算装置は、前記コード暗号鍵を用いて前記保護プログラムを復号する、

ことを含むことを特徴とするプログラム実行許諾方法。

35.    コンピュータにおいて実行されるプログラムコードであって、

前記プログラムコードは、コード暗号鍵によって暗号化され、

15    前記プログラムコードに対応して、前記コード暗号鍵を含み、且つ、前記プログラムコードを実行すべきコンピュータに備えられた中央演算装置が秘密裏に備える秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、

前記ライセンスは前記プログラムコードが実行される前に前記中央演算装置に投入され、

20    前記中央演算装置によって前記秘密鍵を用いて前記ライセンスが復号され、

前記プログラムコードは、前記ライセンスから取得された前記コード暗号鍵を用いて、前記中央演算装置によって復号される、

ことを特徴とするプログラム。

36.    コンピュータにおいて実行されるプログラムコードを記録する、前記

25    コンピュータによって読み取り可能な記録媒体であって、

- 前記プログラムコードは、コード暗号鍵によって暗号化され、
- 前記プログラムコードに対応して、前記コード暗号鍵を含み、且つ、前記プログラムコードを実行すべきコンピュータに備えられた中央演算装置が秘密裏に備える秘密鍵と対になる公開鍵によって暗号化されたライセンスが存在し、
- 5 前記ライセンスは前記プログラムコードが実行される前に前記中央演算装置に投入され、
- 前記中央演算装置によって前記秘密鍵を用いて前記ライセンスが復号され、
- 前記プログラムコードは、前記ライセンスから取得された前記コード暗号鍵を用いて、前記中央演算装置によって復号される、
- 10 ことを特徴とするプログラムを記録する記録媒体。
37. 秘密裏に隠された秘密鍵と、暗号化及び復号を行う暗号手段とを備える中央演算装置を備えるコンピュータにおいて実行されるプログラムを生成するプログラム生成装置であって、
- コードオブジェクトを入力する入力手段と、
- 15 入力された前記コードオブジェクトを複数のブロックに分割し、それぞれのブロックにNOP命令を追加するリンカ前処理手段と、
- アドレス解決を行うリンカ手段と、
- 各ブロックをコード暗号鍵を用いて暗号化することにより保護コード実行形式を生成する保護コード実行形式生成手段と、
- 20 前記コード暗号鍵を含み、前記秘密鍵と対になる公開鍵によって暗号化されたライセンスを生成するライセンス生成手段とを備え、
- 前記ライセンスは、前記コンピュータが前記保護コード実行形式を実行する前に前記中央演算装置に投入され、前記暗号手段によって前記秘密鍵を用いて復号され、
- 25 前記保護コード実行形式は、前記ライセンスから取得された前記コード暗号

鍵を用いて、前記暗号手段によって復号される、  
ことを特徴とするソフトウェア生成装置。



## 要約書

中央演算装置（G T） 1 0 は、秘密裏に秘密鍵と、暗号エンジン 1 6 を備える。G T 1 0 が、ソフトウェアを実行する前には、ソフトウェアに付加されたライセンスが G T 1 0 に投入される。ライセンスは、ソフトウェアを暗号化する際に用いられたコード暗号鍵を、秘密鍵と対になる公開鍵で暗号化した情報を含む。

ライセンスが投入されると、暗号エンジン 1 6 は、秘密鍵を用いてライセンスを復号することによりコード暗号鍵を取得し、そのコード暗号鍵を用いてソフトウェアを復号する。

1 / 5 2

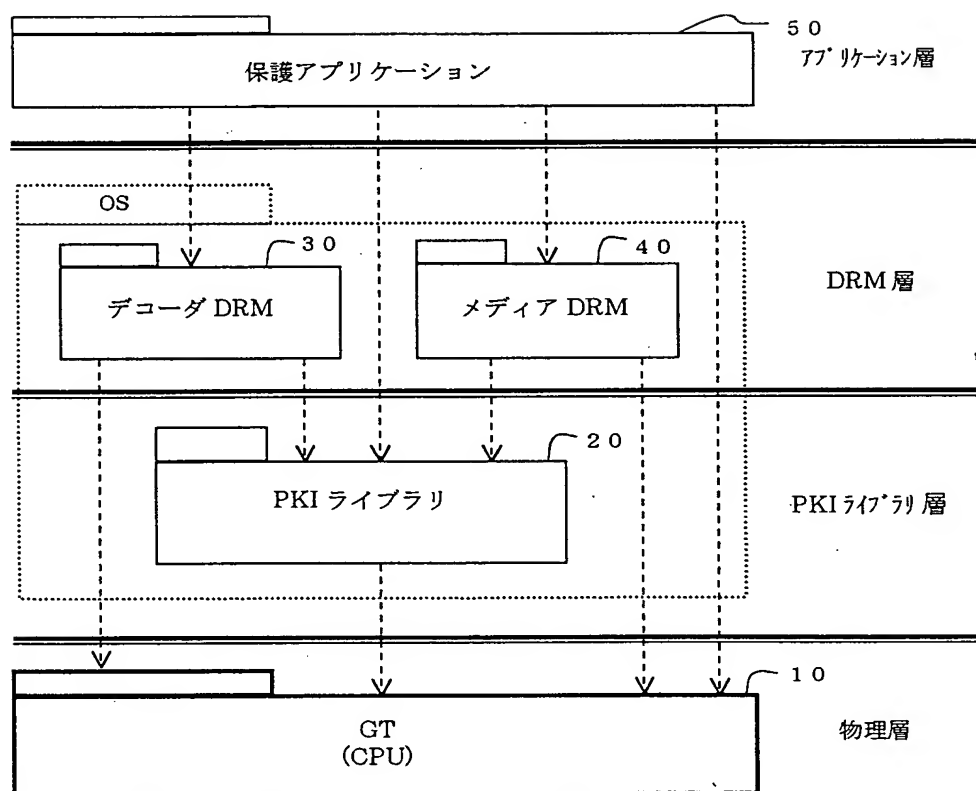


図 1

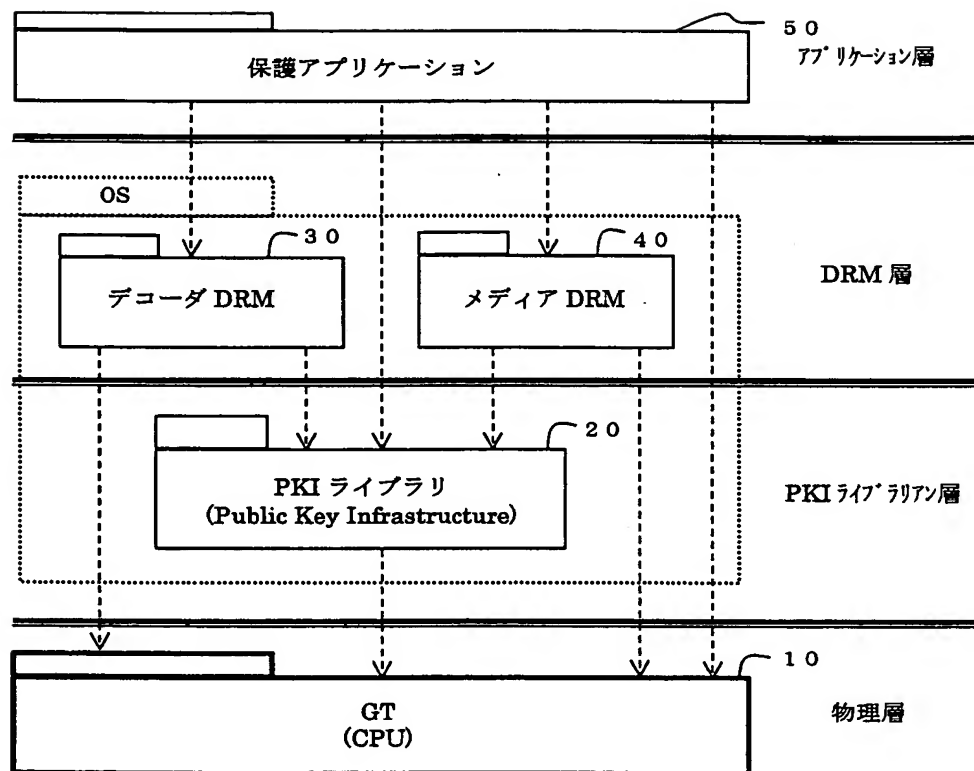


図 1

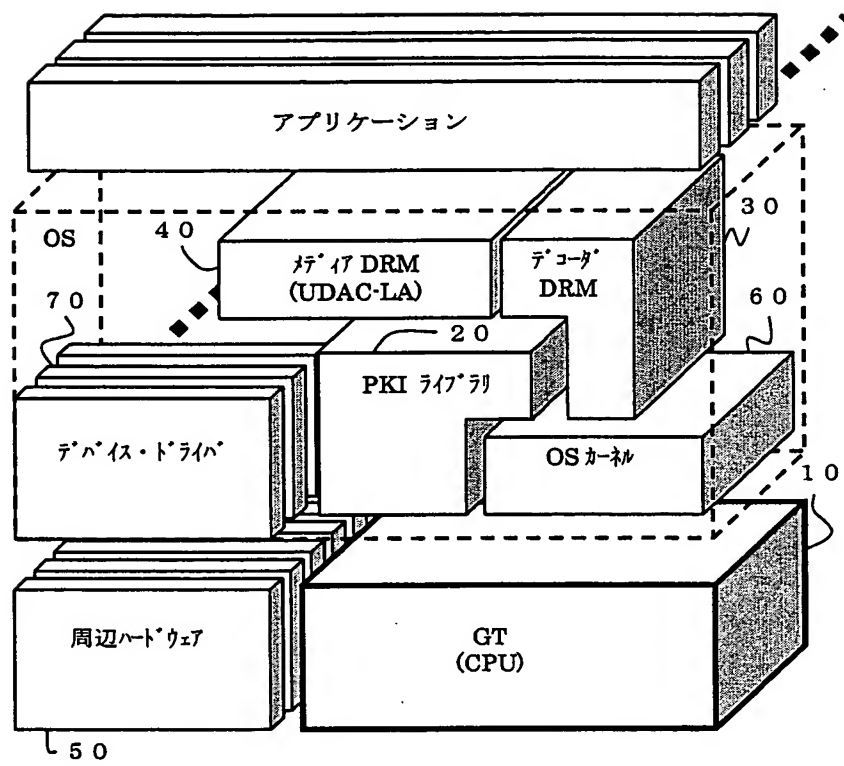


図 2

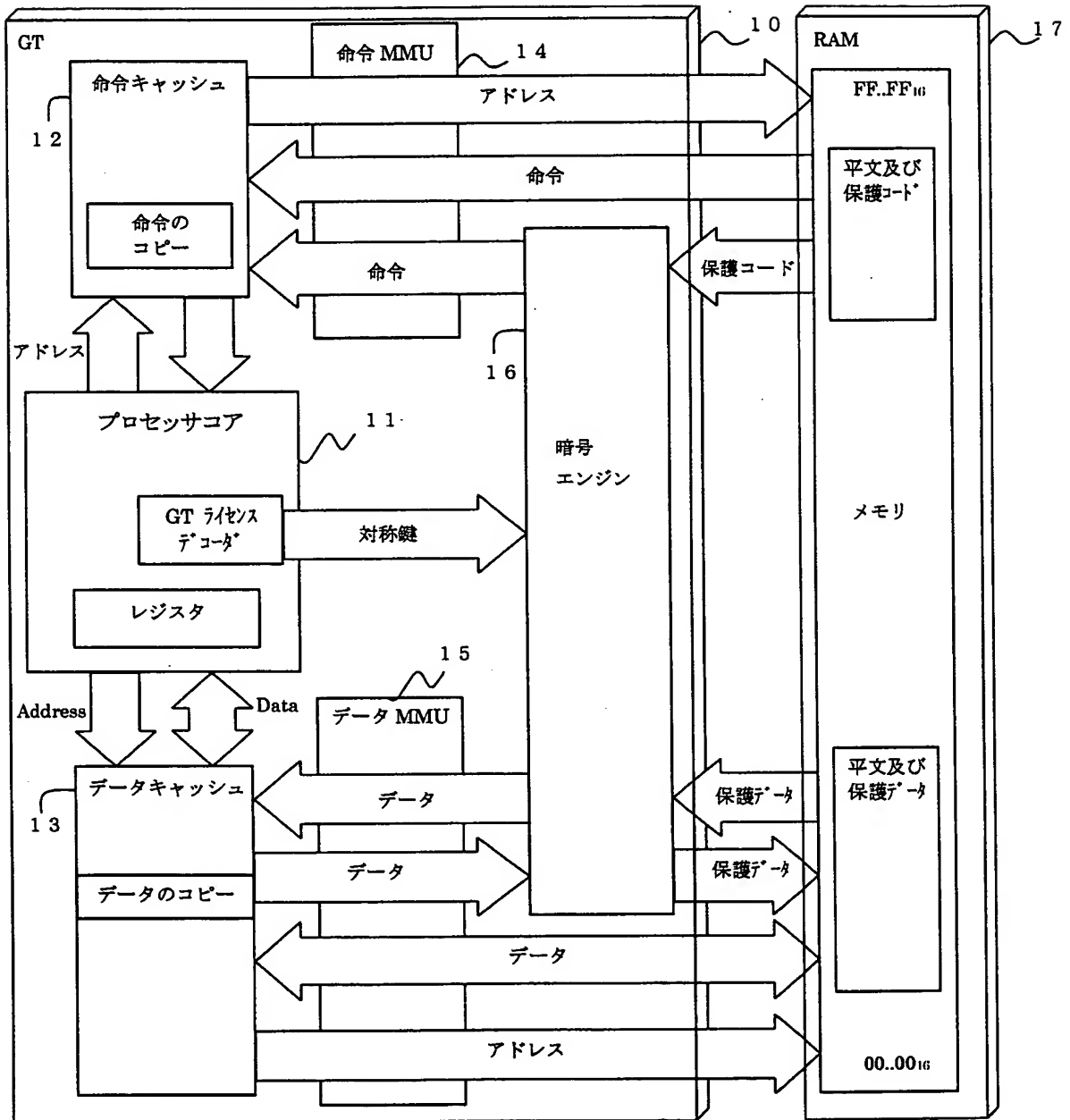


図 3

| Notation | Field Size | Name                           | Description                                            |
|----------|------------|--------------------------------|--------------------------------------------------------|
| p        | 1 bit      | Present flag<br>有効性フラグ         | on(1)であれば、TRBが有効であることを示す。                              |
| uo       | 1 bit      | Unable to Output<br>外部出力禁止フラグ  | on(1)であれば、暗号化キーテーブルに出力されないことを示す。                       |
| cl       | 1 bit      | Cache Lock<br>キャッシュロックフラグ      | on(1)であれば、kidが指定された耐攻撃ページのデータがキャッシュの外に出力されないことを示す。     |
| kid      | 20 bit程度   | Key ID<br>鍵識別情報                | TLBからのリンクのための情報。                                       |
| key      | 128 bit以上  | Cryptographic Key<br>暗号鍵       | このラインにリンクするページのコードまたはデータの暗号・復号鍵の値。対称鍵（共通鍵）暗号法の鍵        |
| ackey    | keyに同じ     | Authorized code key<br>被許諾コード鍵 | kidを含む全TLB行のページにアクセスが許可された保護プロセスの実行コード復号鍵(他行又は自行のkey)。 |
| eadr     | 64 bit程度   | Exception address<br>例外処理アドレス  | keyが異なるページから、このTRBにリンクしたページに復帰する直前に、発生する例外処理コードの先頭アドレス |

| Notation |    | Field Size | Name                                          | Description                                                    |
|----------|----|------------|-----------------------------------------------|----------------------------------------------------------------|
| p        |    | 1 bit      | Present flag<br>有効性フラグ                        | TLBが有効であることを示す。                                                |
| id       |    | 20 bit程度   | Region Identifier<br>領域識別子                    | TLB内の当該行が示すページ領域の識別子値。                                         |
| ppn      |    | 32 bit程度   | Physical page number<br>物理ページ番号               | 従来からのTLBの物理ページ番号。                                              |
| vpn      |    | 56 bit程度   | Virtual page number<br>仮想ページ番号                | 従来からのTLBの仮想ページ番号。                                              |
| rights   | pl | 2 bit      | Privilege level<br>権限レベル                      | ページにアクセス可能な権限のレベル。詳細は図6及び図7のとおり。                               |
|          | ar | 3 bit      | Access Rights<br>アクセス権                        | ページへのアクセス権の種類を指定する。詳細は図6及び図7のとおり。                              |
|          | tr | 1 bit      | Tamper Resistance<br>耐攻撃性フラグ                  | on(1)であれば、耐攻撃セグメント空間にあり、TRBを持つことを示す。                           |
|          | df | 1 bit      | Debug mode Flag<br>デバッグモードフラグ                 | trがon(1)の場合のみ有効。trとdfがonならar指定ごとのデバッグモードで動作する。動作詳細は3. 1. 2を参照。 |
| kid      |    | 20 bit程度   | Key ID<br>鍵識別情報                               | TRB内の鍵の情報にリンクするためのTRB行(insertion)識別情報。                         |
| ebim     |    | 3 bit程度    | Encrypted Block Identification<br>暗号化ブロック識別方式 | GTライセンスのACgt.ebimの値のコピー。trがoff(0)のときにはこの値も0となる。                |
| sign     |    | 128 bit程度  | Digital signature<br>デジタル署名                   | vpnからebimまでのフィールド連結データのデジタル署名値。GTが生成。                          |

図5

| PSR.S<br>実行モード          | PTE.TR<br>(TLB.tr) | PTE.PP<br>(TLB.ar) | IF (命令<br>フェッチ) | Load (ロー<br>ド命令) | Store (スト<br>ア命令) |
|-------------------------|--------------------|--------------------|-----------------|------------------|-------------------|
| 1<br>スーパー<br>バイザ<br>モード | 0<br>一般            | 000                | OK              | OK               | OK                |
|                         |                    | 001                | OK              | OK               | NG                |
|                         |                    | 010                | OK              | NG               | NG                |
|                         |                    | 011                | NG              | NG               | NG                |
|                         | 1<br>耐攻撃           | 000                | 指定コード           | 指定コード            | 指定コード             |
|                         |                    | 001                | 指定コード           | 指定コード            | NG                |
|                         |                    | 010                | 指定コード           | NG               | NG                |
|                         |                    | 011                | NG              | NG               | NG                |
| 0<br>ユーザ<br>モード         | 0<br>一般            | 100                | OK              | OK               | OK                |
|                         |                    | 101                | OK              | OK               | NG                |
|                         |                    | 110                | OK              | NG               | NG                |
|                         |                    | 111                | NG              | NG               | NG                |
|                         | 1<br>耐攻撃           | 100                | 指定コード           | 指定コード            | 指定コード             |
|                         |                    | 101                | 指定コード           | 指定コード            | NG                |
|                         |                    | 110                | 指定コード           | NG               | NG                |
|                         |                    | 111                | NG              | NG               | NG                |



| TLB.rights          |    | TLB.pl | Process Privilege Level |     |     |     | Description                       |
|---------------------|----|--------|-------------------------|-----|-----|-----|-----------------------------------|
| tr & df             | ar |        | 3                       | 2   | 1   | 0   |                                   |
| tr=0<br>or<br>tr=1, | 0  | 3      | R                       | R   | R   | R   | read のみ                           |
|                     |    | 2      | -                       | R   | R   | R   |                                   |
|                     |    | 1      | -                       | -   | R   | R   |                                   |
|                     |    | 0      | -                       | -   | -   | R   |                                   |
|                     | 1  | 3      | RX                      | RX  | RX  | RX  | read, execute                     |
|                     |    | 2      | -                       | RX  | RX  | RX  |                                   |
|                     |    | 1      | -                       | -   | RX  | RX  |                                   |
|                     |    | 0      | -                       | -   | -   | RX  |                                   |
|                     | 2  | 3      | RW                      | RW  | RW  | RW  | read, write                       |
|                     |    | 2      | -                       | RW  | RW  | RW  |                                   |
|                     |    | 1      | -                       | -   | RW  | RW  |                                   |
|                     |    | 0      | -                       | -   | -   | RW  |                                   |
|                     | 3  | 3      | RWX                     | RWX | RWX | RWX | read, write, execute              |
|                     |    | 2      | -                       | RWX | RWX | RWX |                                   |
|                     |    | 1      | -                       | -   | RWX | RWX |                                   |
|                     |    | 0      | -                       | -   | -   | RWX |                                   |
| tr=1,<br>df=0       | 0  | 3      | PR                      | PR  | PR  | PR  | 指定プロセスからのreadのみ                   |
|                     |    | 2      | -                       | PR  | PR  | PR  |                                   |
|                     |    | 1      | -                       | -   | PR  | PR  |                                   |
|                     |    | 0      | -                       | -   | -   | PR  |                                   |
|                     | 1  | 3      | X                       | X   | X   | X   | 指定プロセスからの<br>execute, read        |
|                     |    | 2      | -                       | X   | X   | X   |                                   |
|                     |    | 1      | -                       | -   | X   | X   |                                   |
|                     |    | 0      | -                       | -   | -   | X   |                                   |
|                     | 2  | 3      | PRW                     | PRW | PRW | PRW | 指定プロセスからの<br>read, write          |
|                     |    | 2      | -                       | PRW | PRW | PRW |                                   |
|                     |    | 1      | -                       | -   | PRW | PRW |                                   |
|                     |    | 0      | -                       | -   | -   | PRW |                                   |
|                     | 3  | 3      | PWX                     | PWX | PWX | PWX | 指定プロセスからの<br>read, write, execute |
|                     |    | 2      | -                       | PWX | PWX | PWX |                                   |
|                     |    | 1      | -                       | -   | PWX | PWX |                                   |
|                     |    | 0      | -                       | -   | -   | PWX |                                   |

図 7

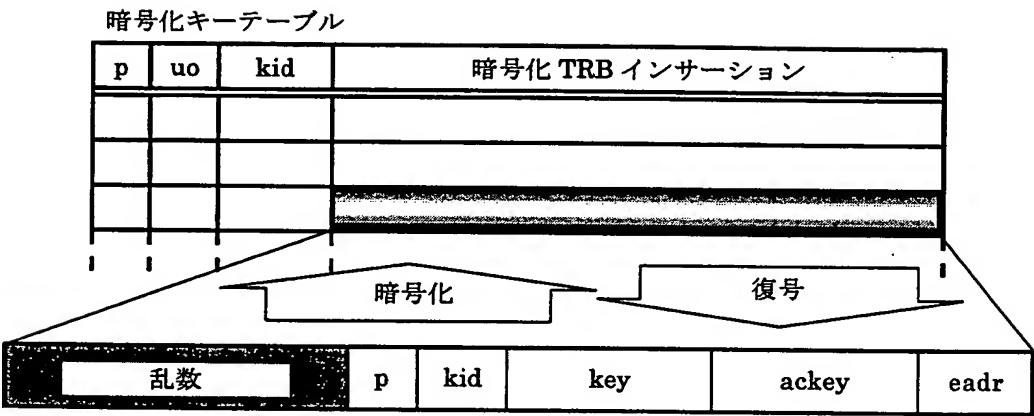


図 8

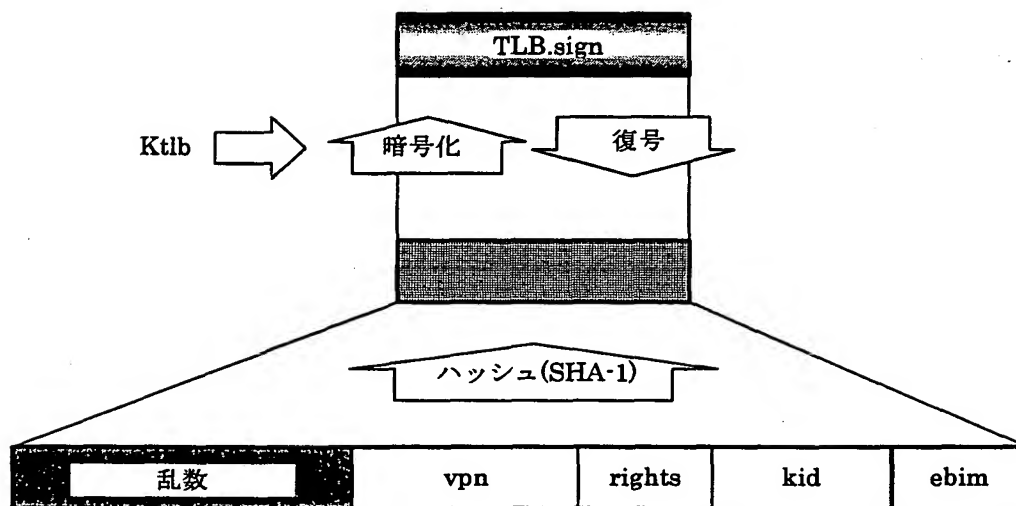


図 9

10/52

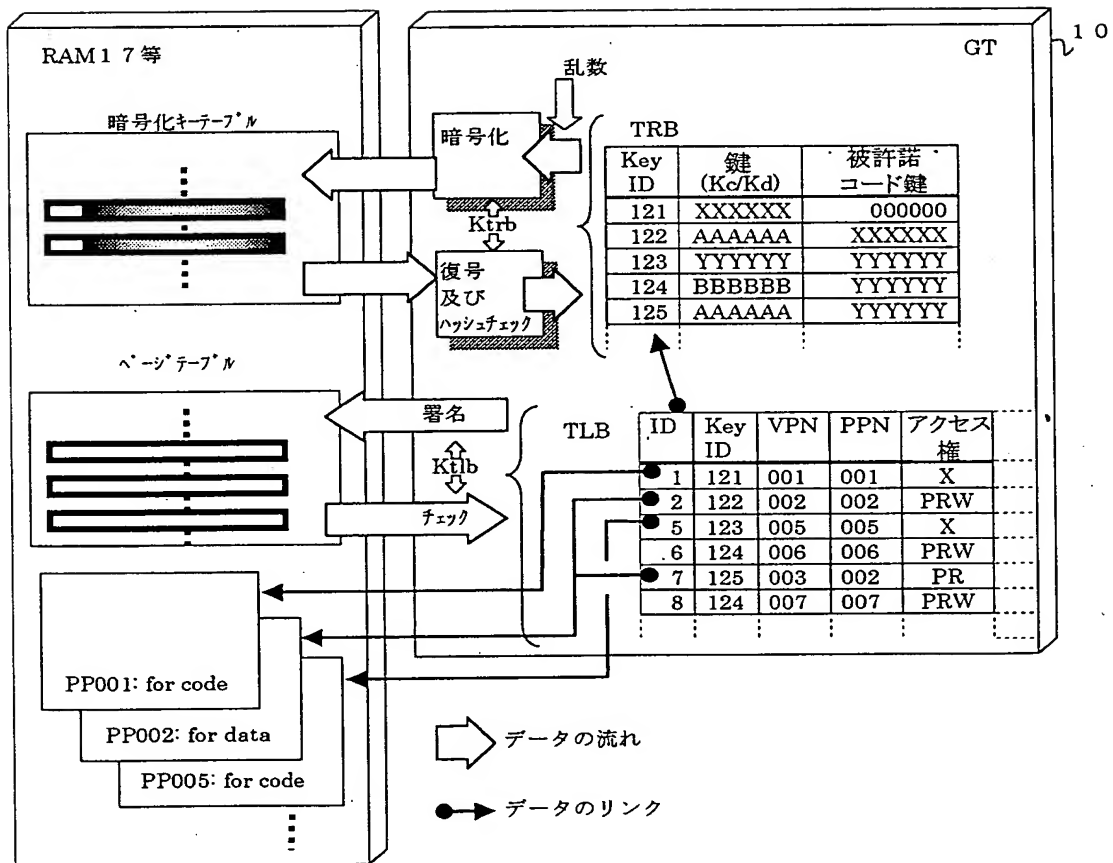


図 10

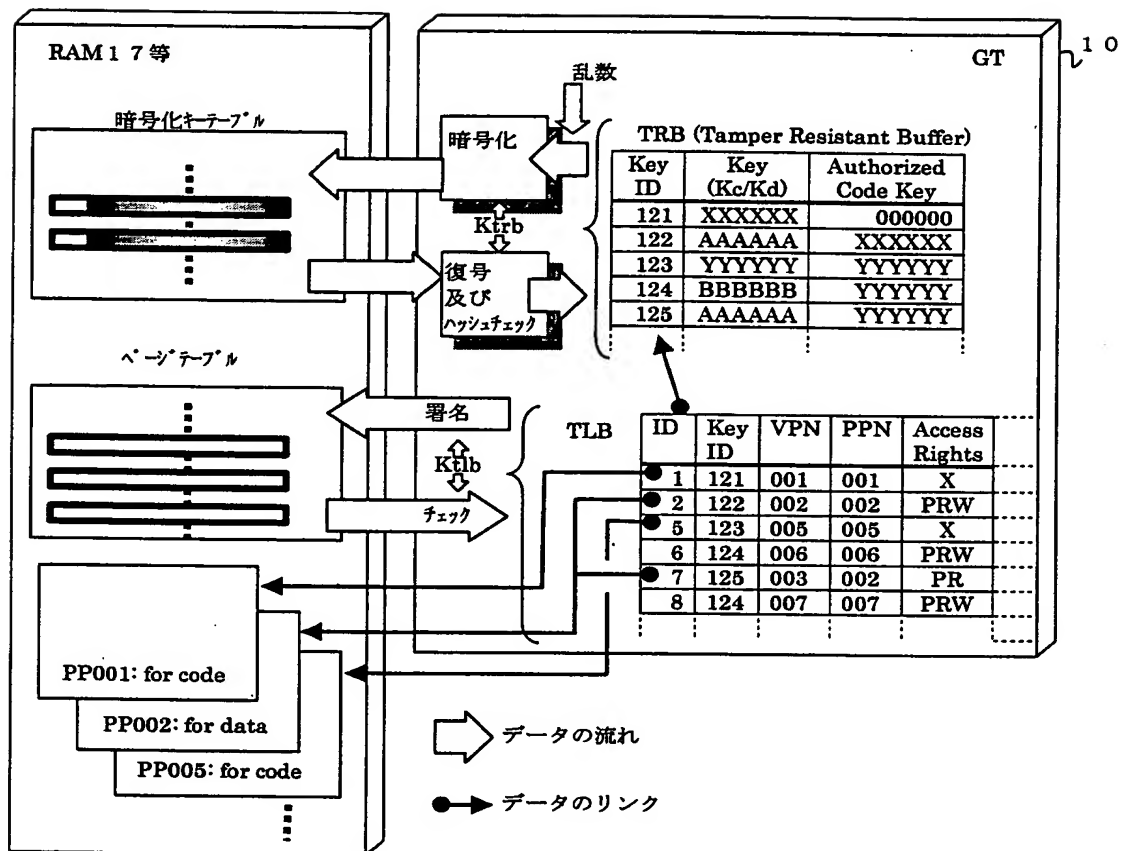


図 10

| Notation | Field Size | Name                                                     | Description                          |
|----------|------------|----------------------------------------------------------|--------------------------------------|
| trss     | 1 bit      | Tamper Resistant<br>Segment Selector<br>耐攻撃セグメントセ<br>レクタ | onで耐攻撃空間が選択され，offで一般<br>の仮想空間が選択される。 |

| Notation | Field Size     | Name                           | Description                                |
|----------|----------------|--------------------------------|--------------------------------------------|
| rid      | 8 bit程度        | Register ID                    | レジスタを指定するID                                |
| seal     | 1 bit          | Seal flag<br>封印フラグ             | on(1)で封印中であることを示し,<br>off(0)で解放されていることを示す。 |
| ackey    | TRB.key<br>に同じ | Authorized code key<br>被許諾コード鍵 | レジスタへのアクセスが許可されて<br>いるプロセスのコードページのKey      |

| Notation | Field Size | Name                                   | Description                                    |
|----------|------------|----------------------------------------|------------------------------------------------|
| rlss     | 1 bit      | Register 1 Seal<br>Status<br>レジスタ1封印状態 | on(1)なら, r1が封印中であることを示し, off(0)なら解除されていることを示す。 |
| r2ss     | 1 bit      | Register 2 Seal<br>Status<br>レジスタ2封印状態 | on(1)なら, r2が封印中であることを示し, off(0)なら解除されていることを示す。 |



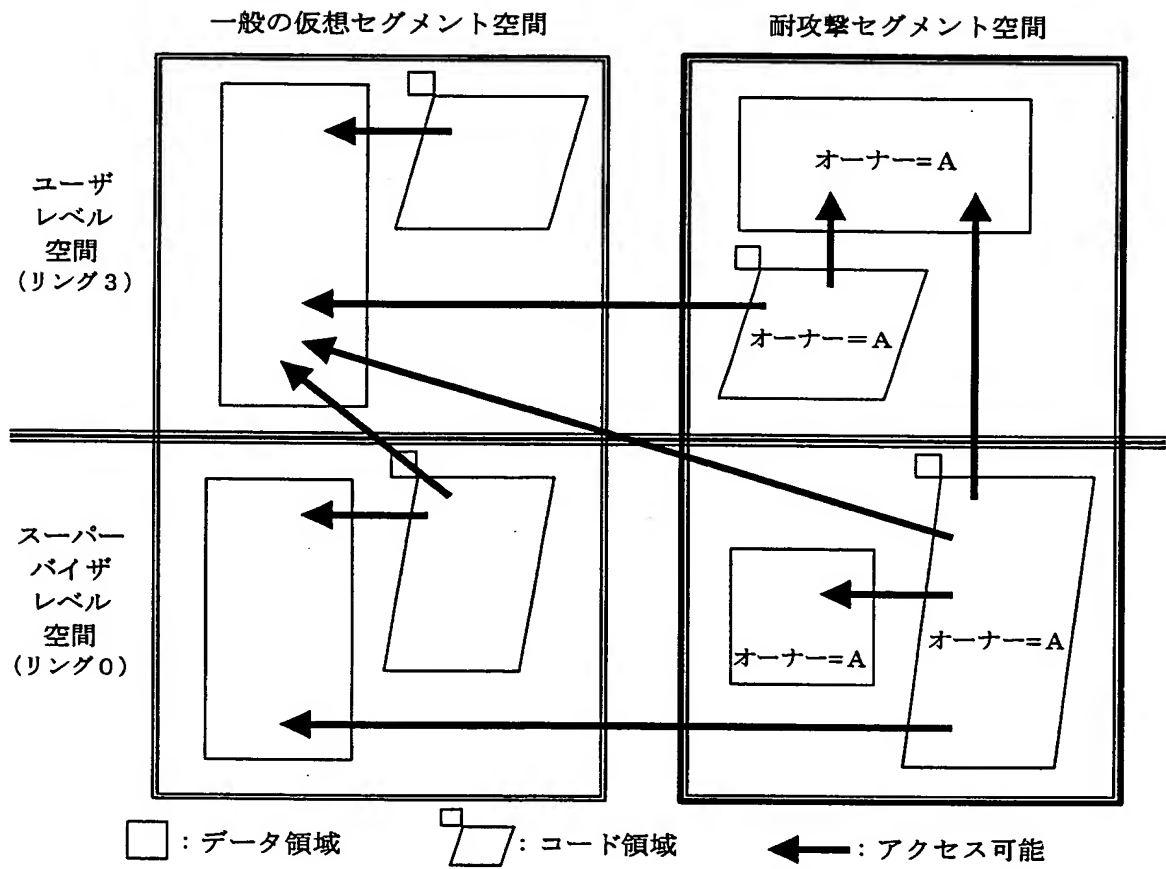


図 14

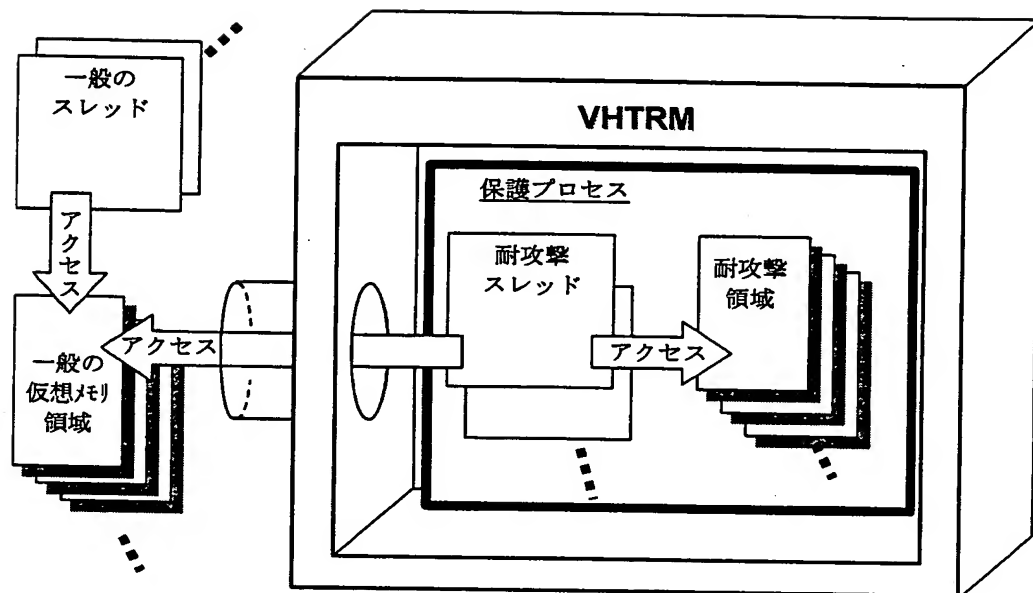


図 15

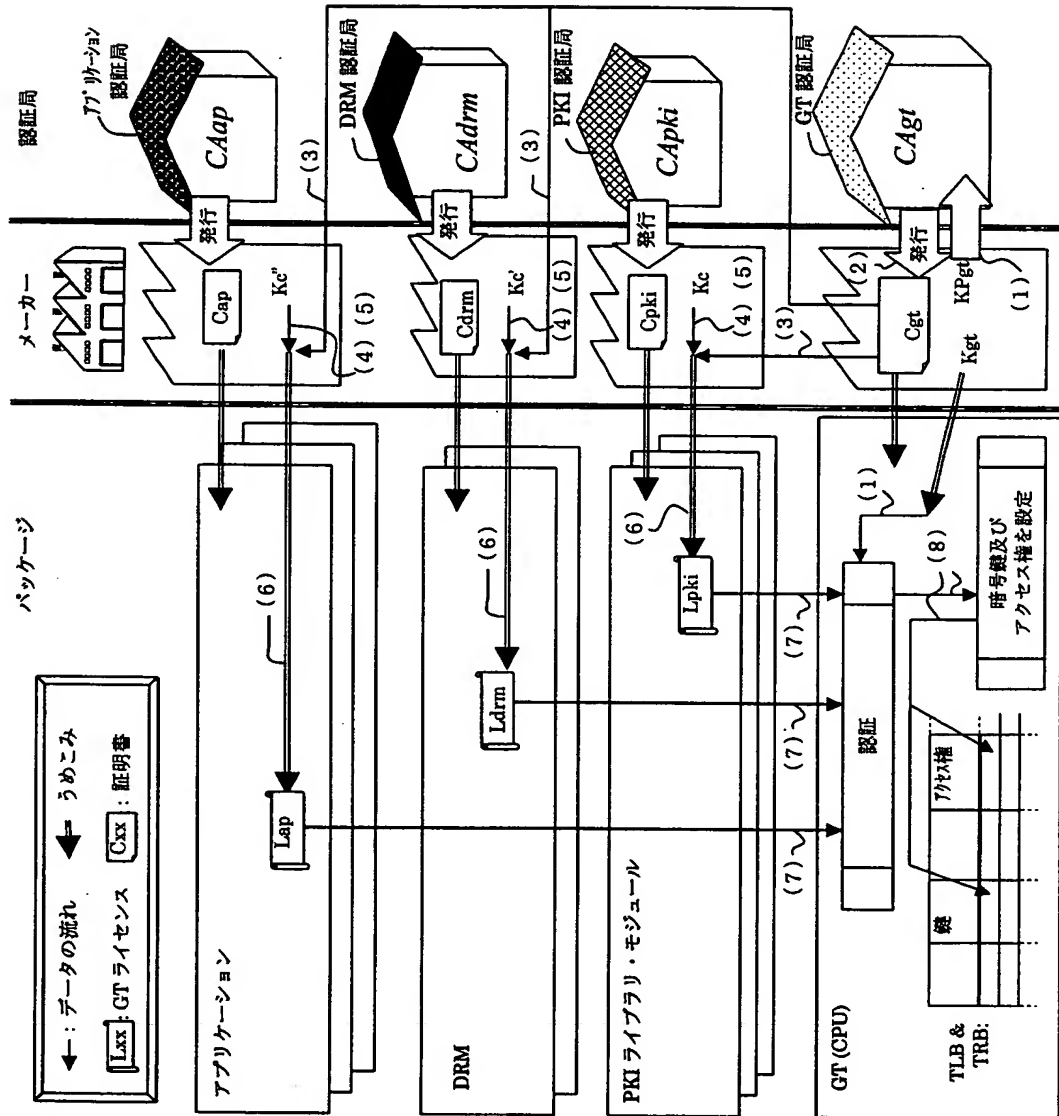


図 16

17 / 52

| 記号            | 意味                 | read権   | write権  | execute権 |
|---------------|--------------------|---------|---------|----------|
| PR            | 指定コードのみ読み出し可       | 指定コード*1 | なし      | なし       |
| X<br>(PRX)    | 指定コードからの読み込みと実行が可能 | 指定コード*1 | なし      | 指定コード*1  |
| PRW           | 指定コードのみ読み書き可       | 指定コード*1 | 指定コード*1 | なし       |
| PWX<br>(PRWX) | 指定コードからの読み書きと実行が可能 | 指定コード*1 | 指定コード*1 | 指定コード*1  |

図 17

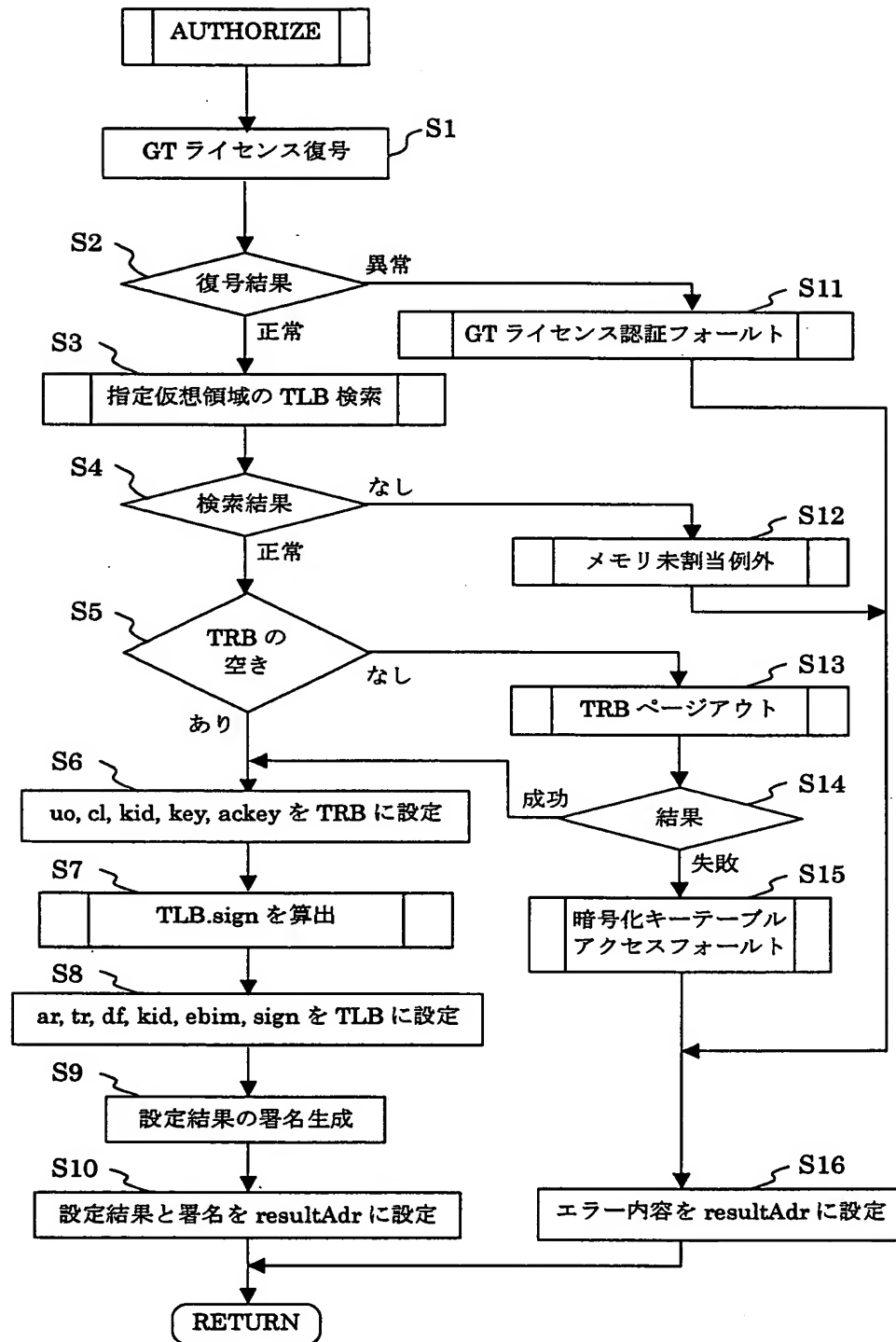


図 18

19 / 52

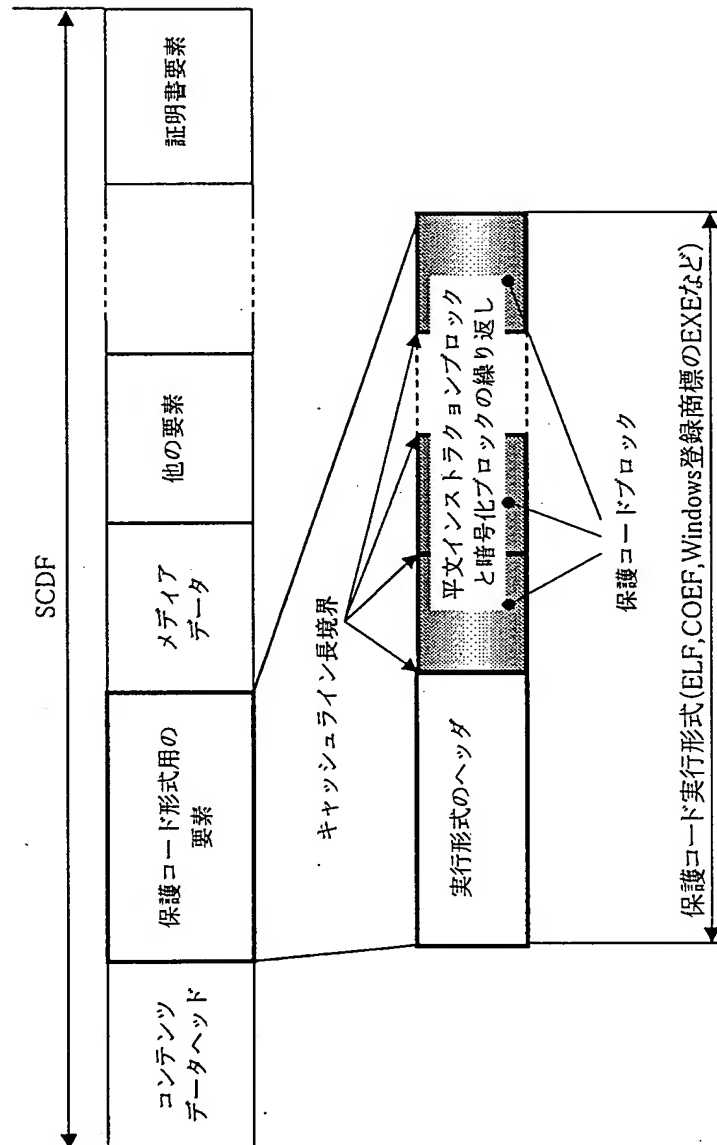


図 19

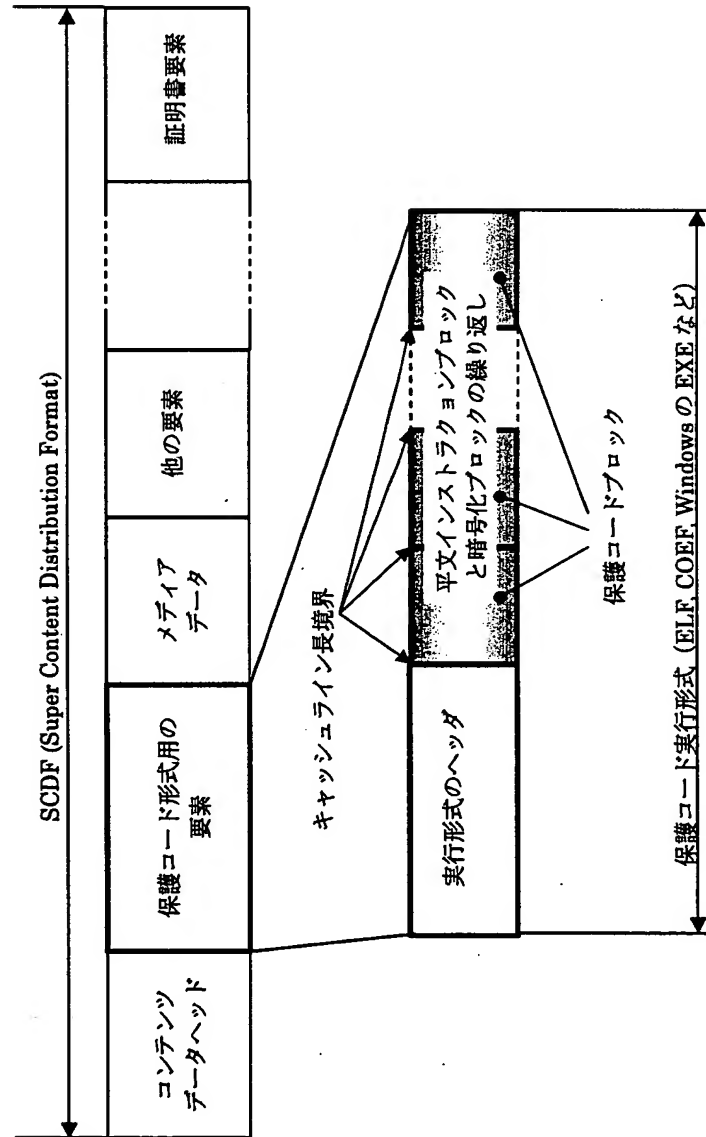


図 19

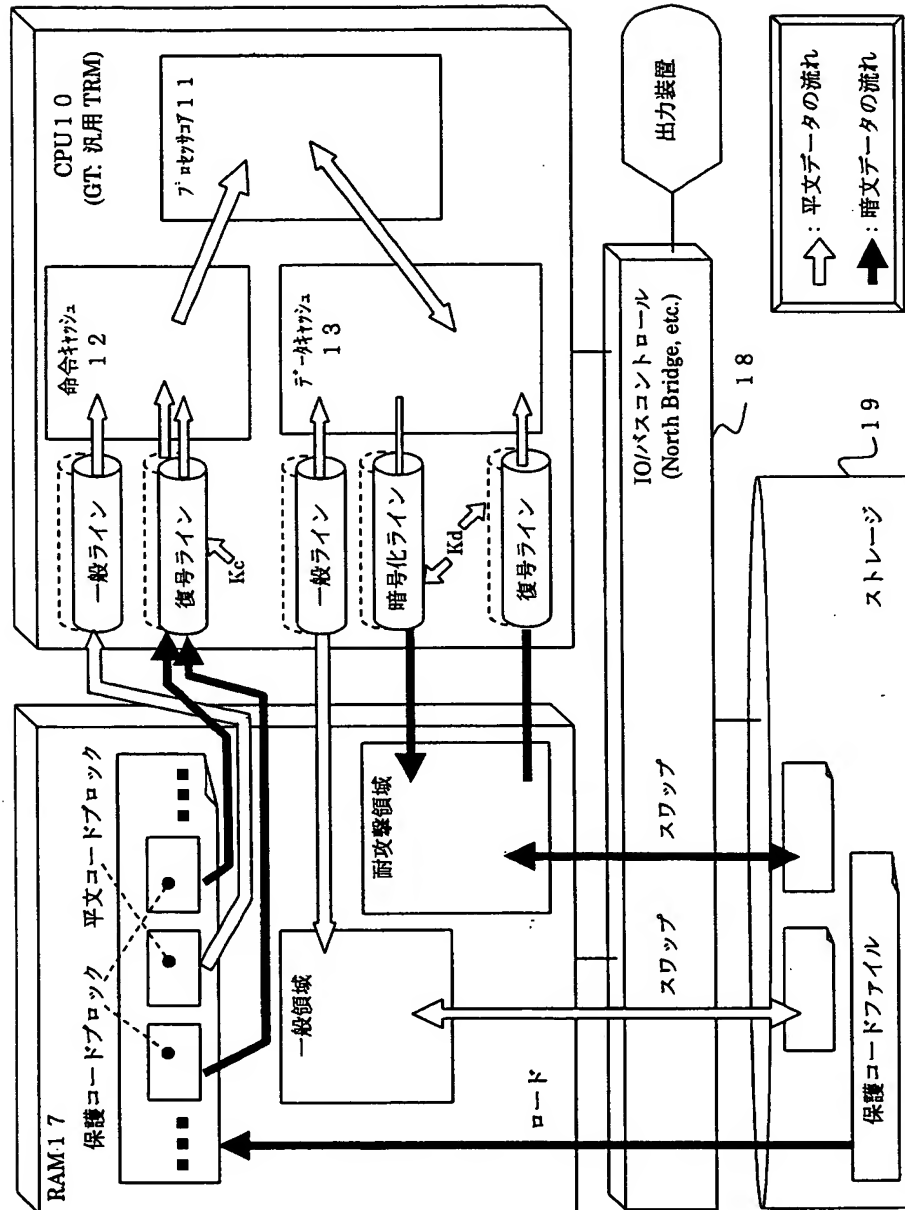


図 20



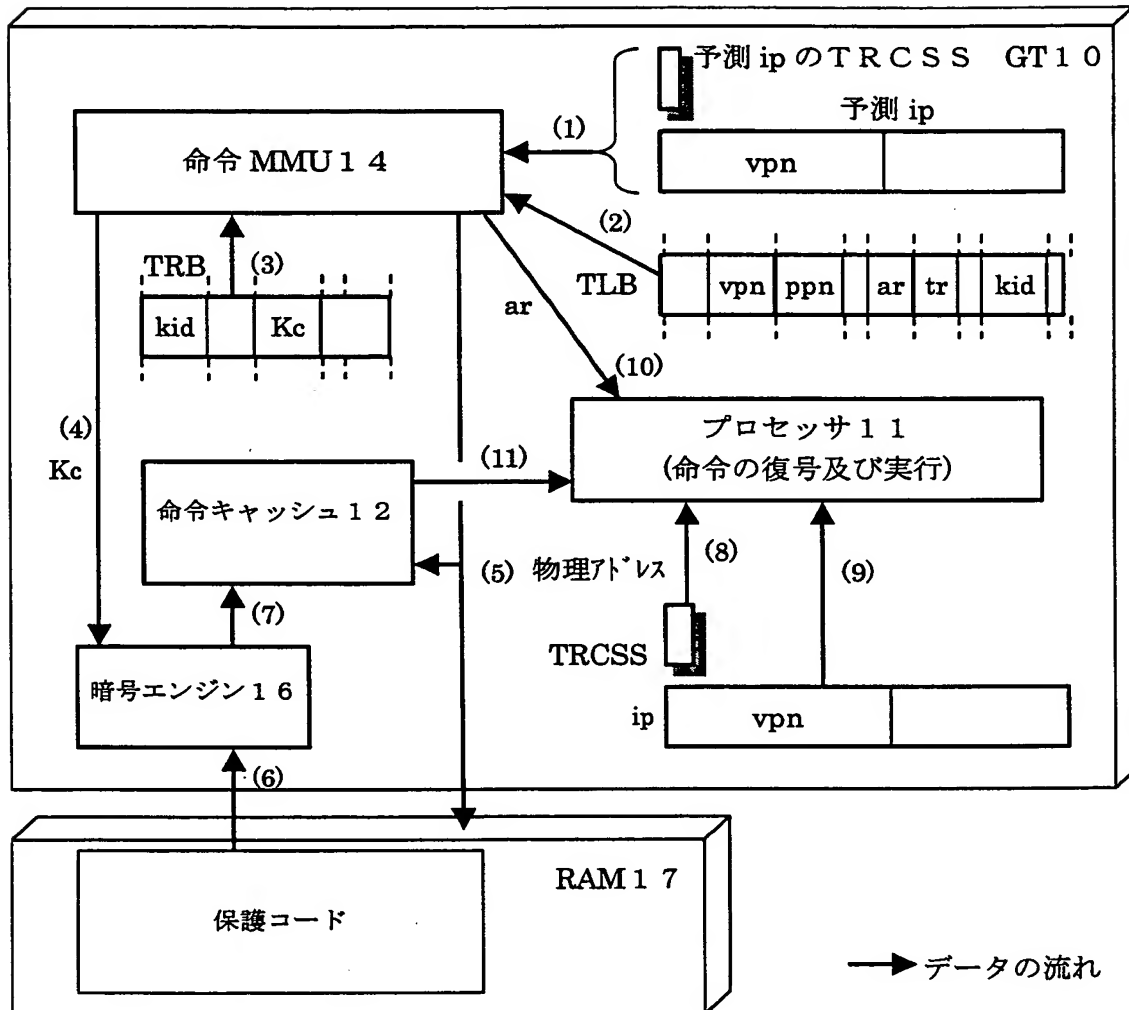


図 2 1

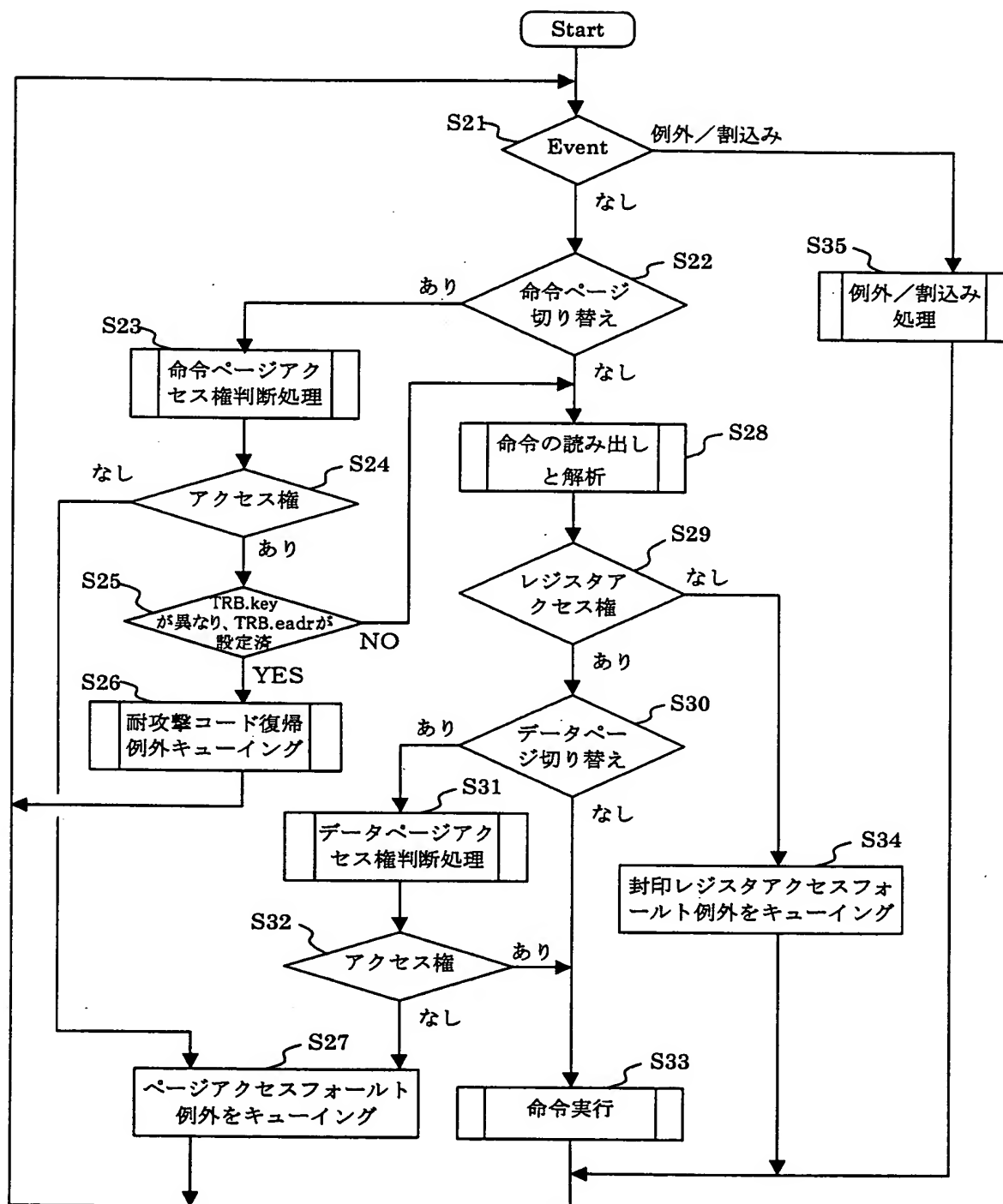


図 22

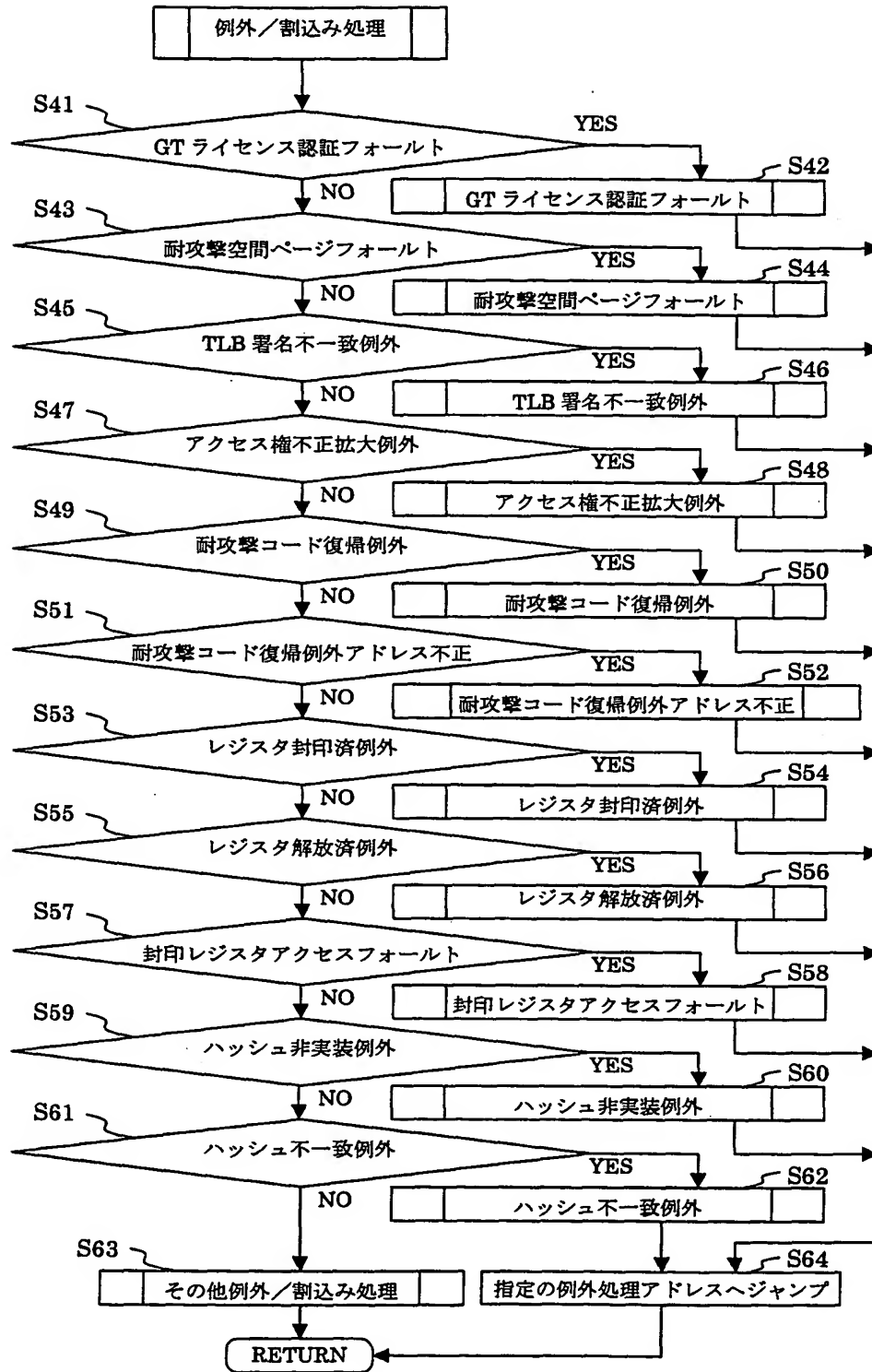


図 23

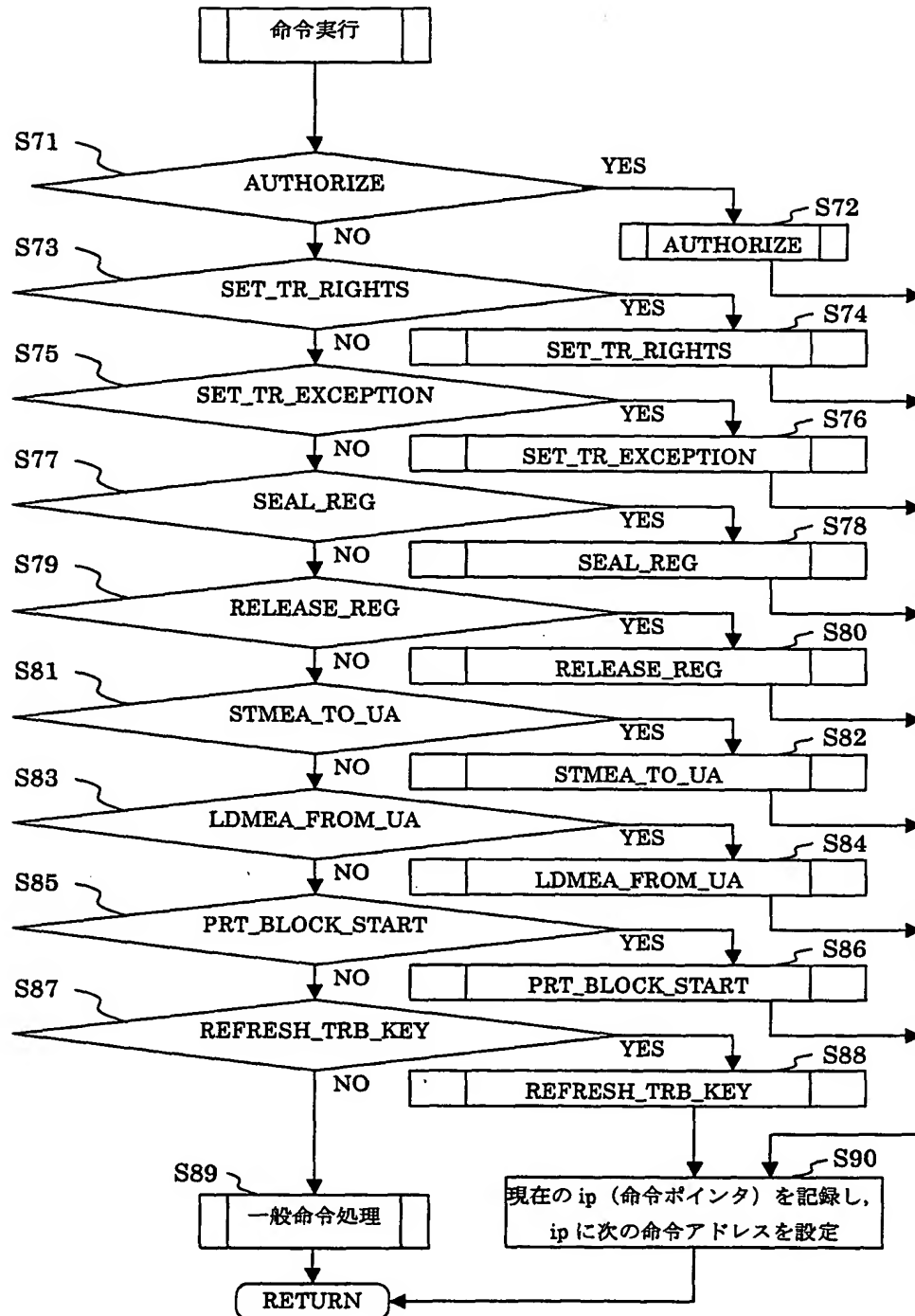


図 24

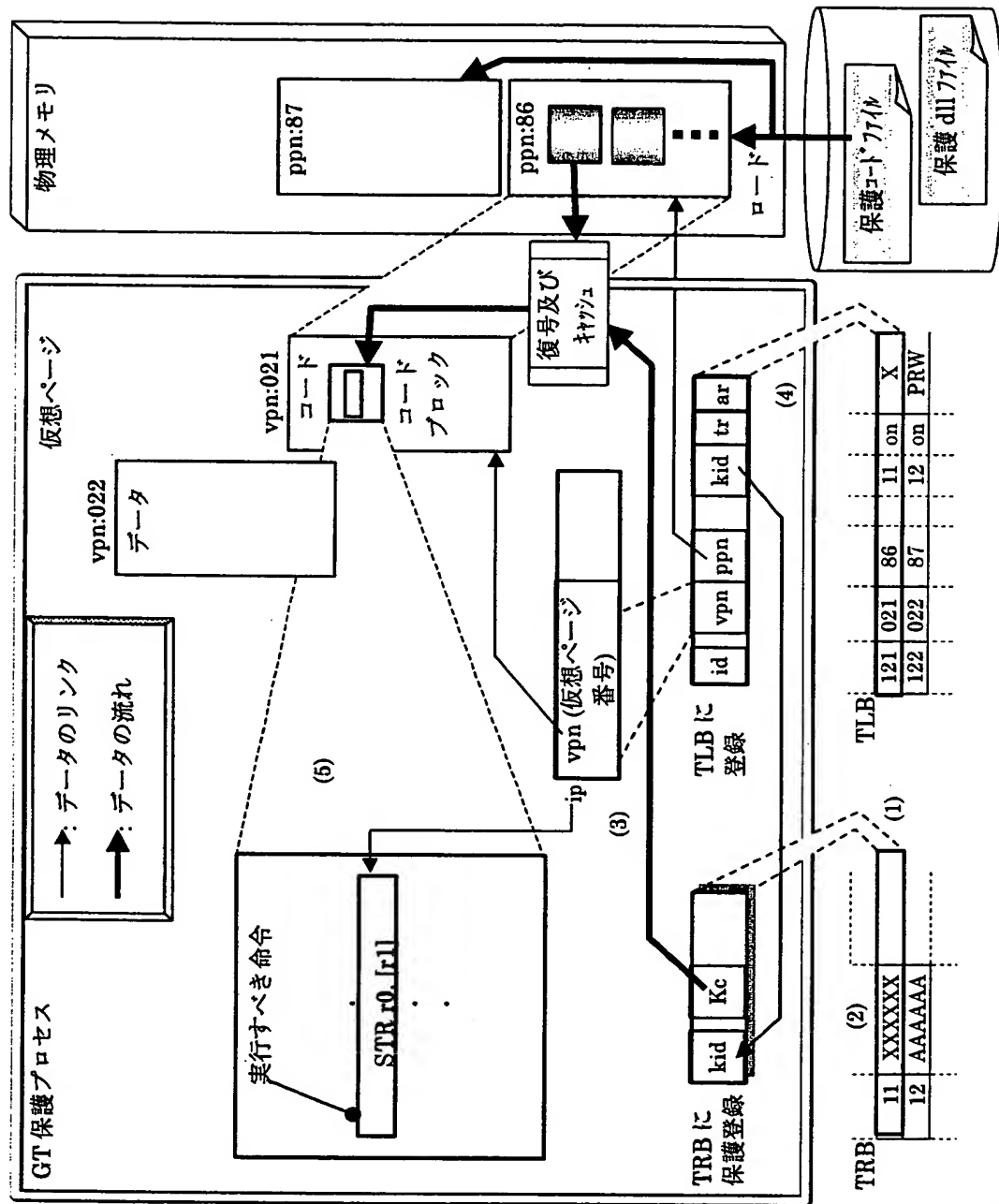


図 25

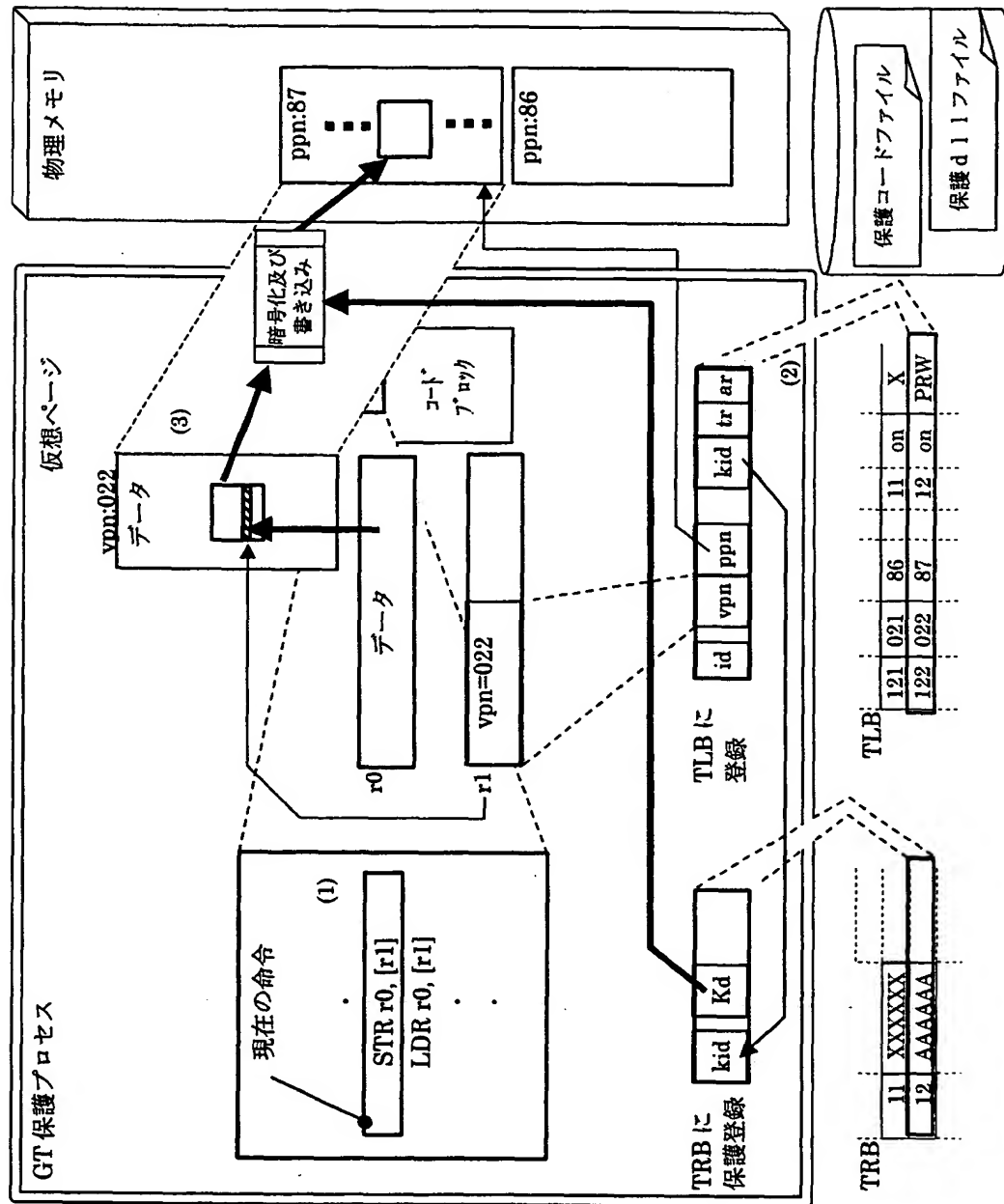


図 26

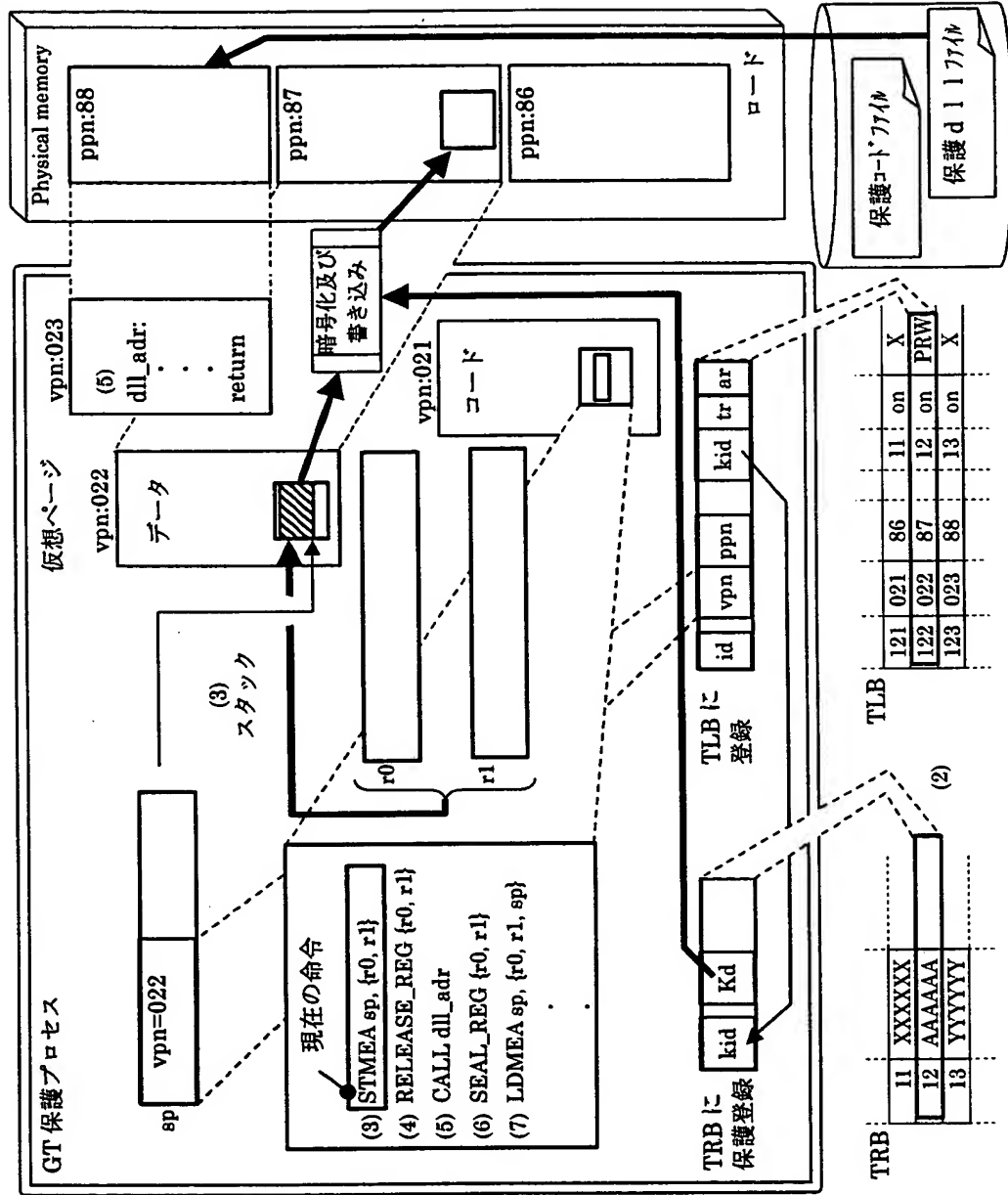


図 27

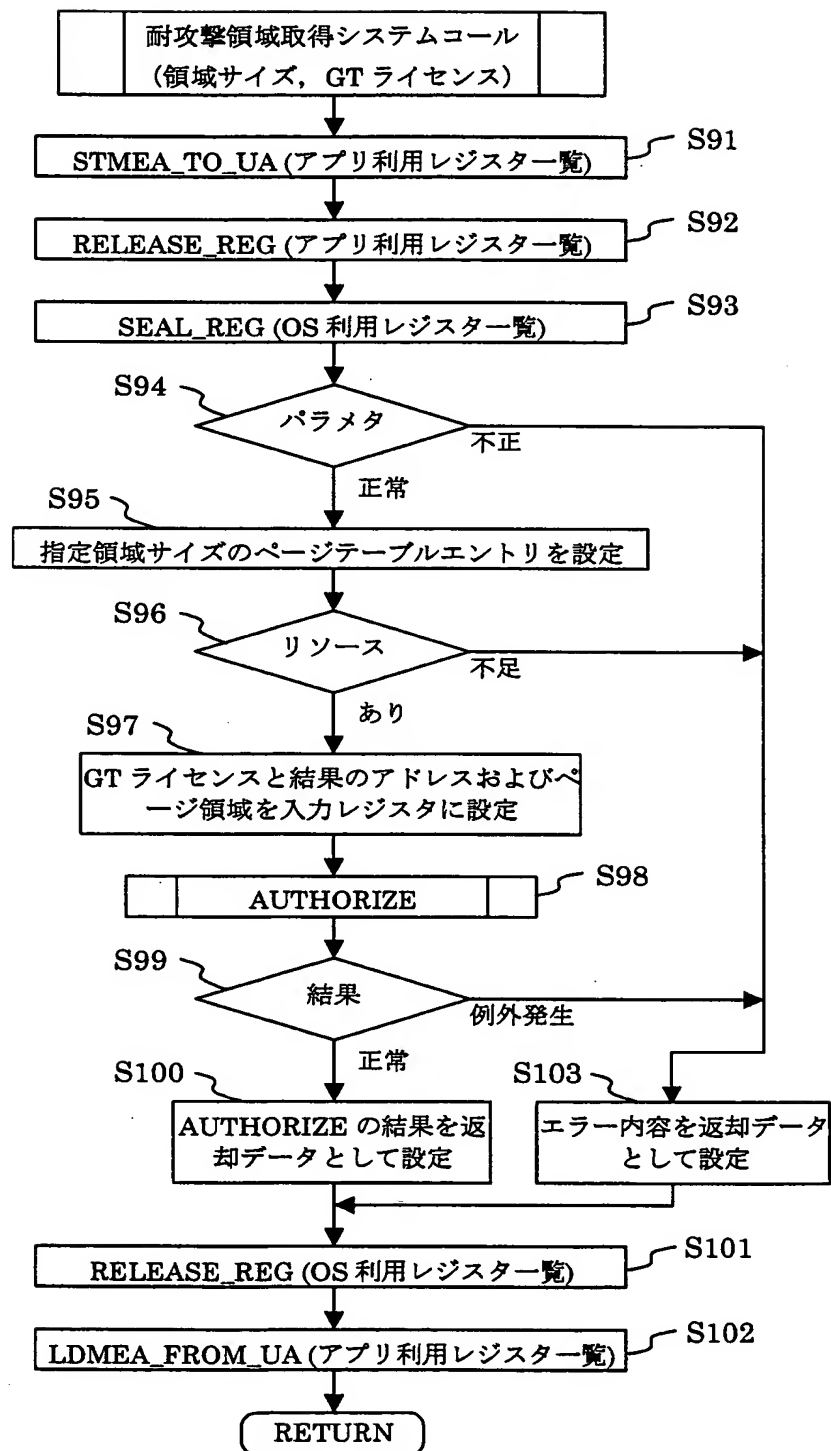


図 28



29 / 52

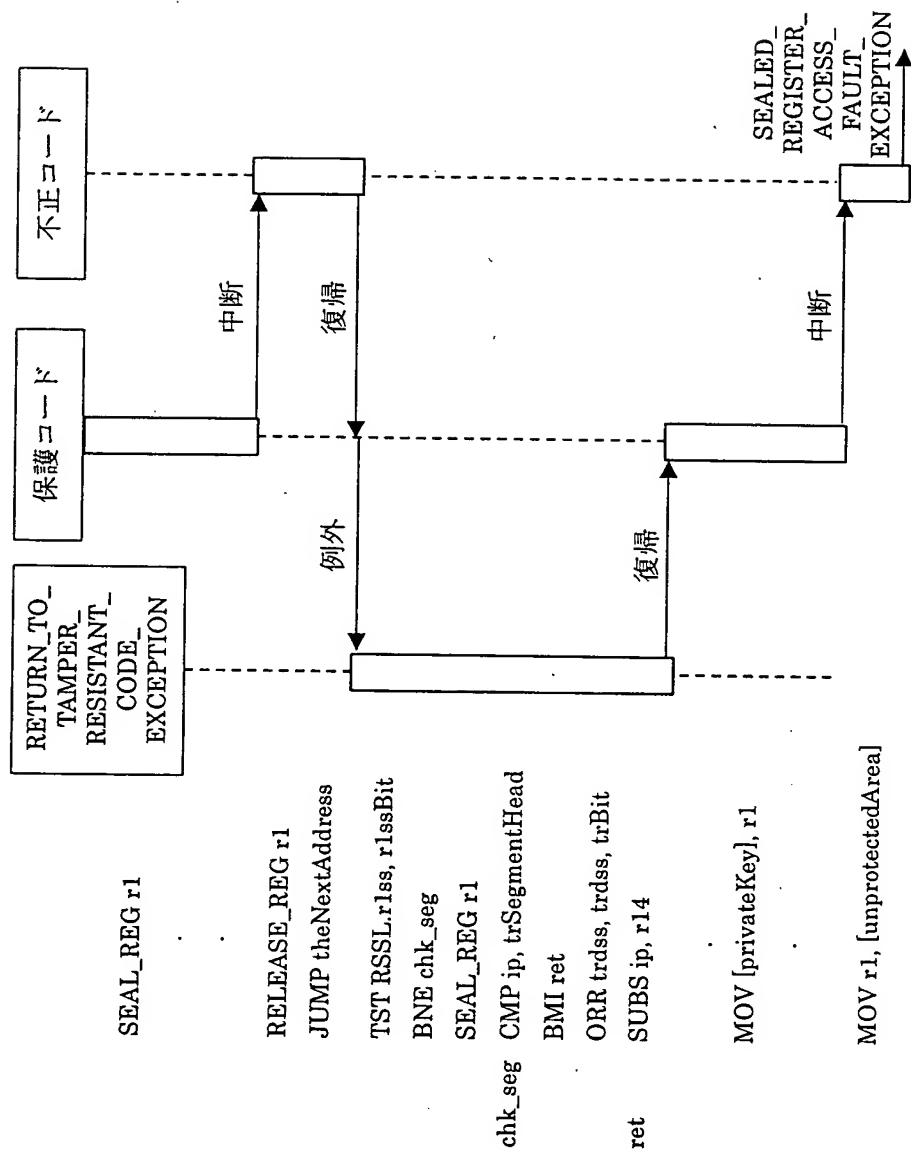


図 29



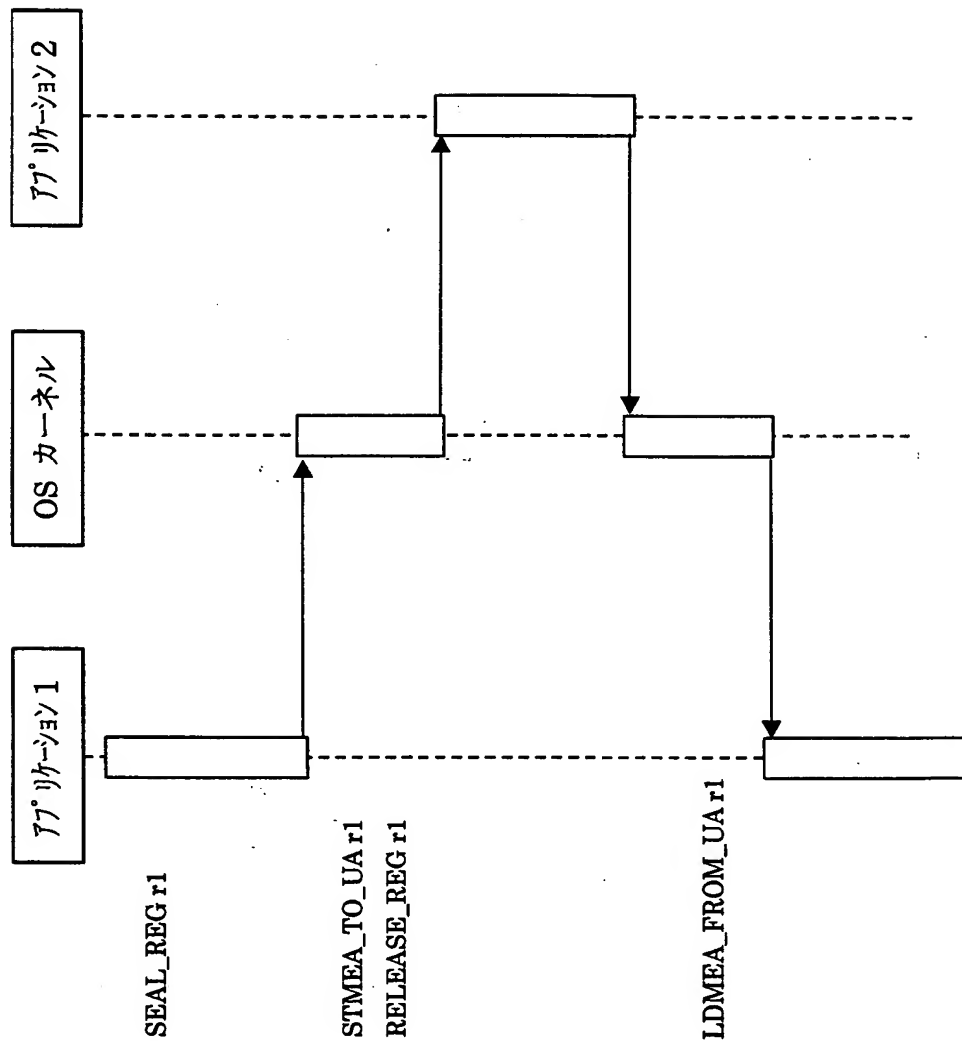


図 30

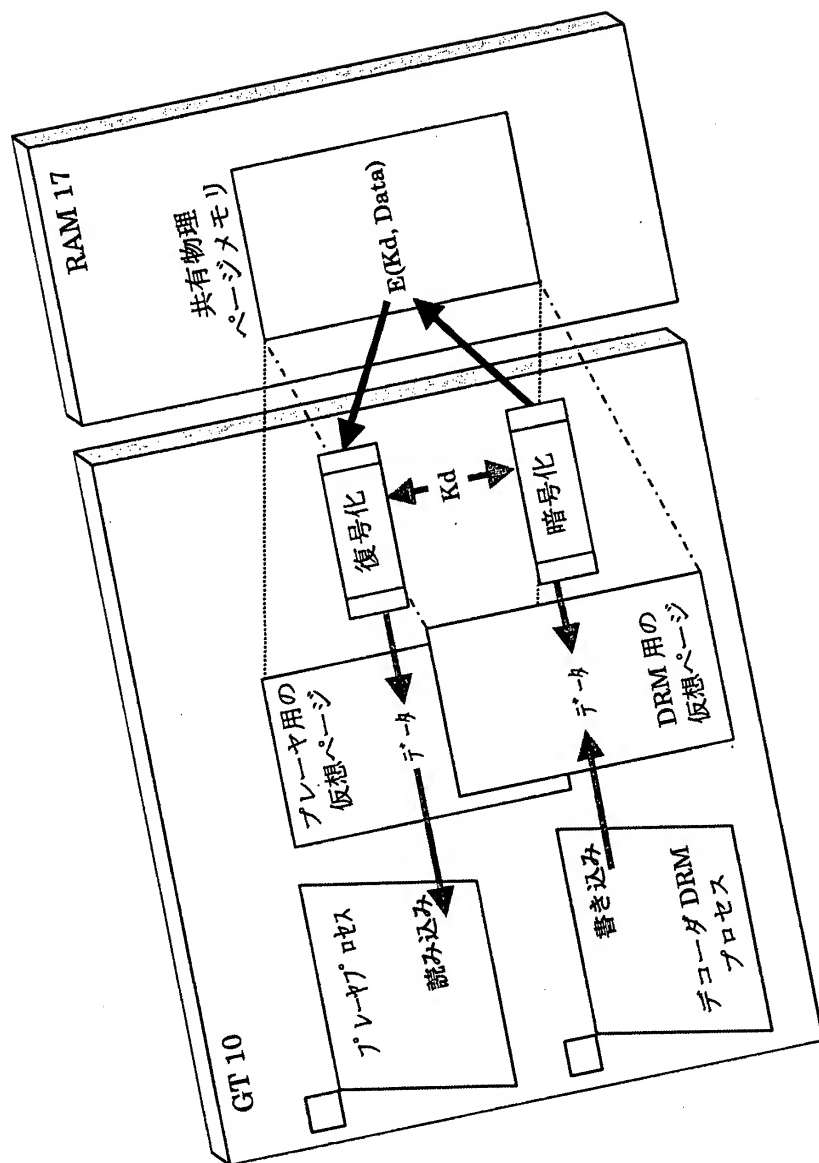
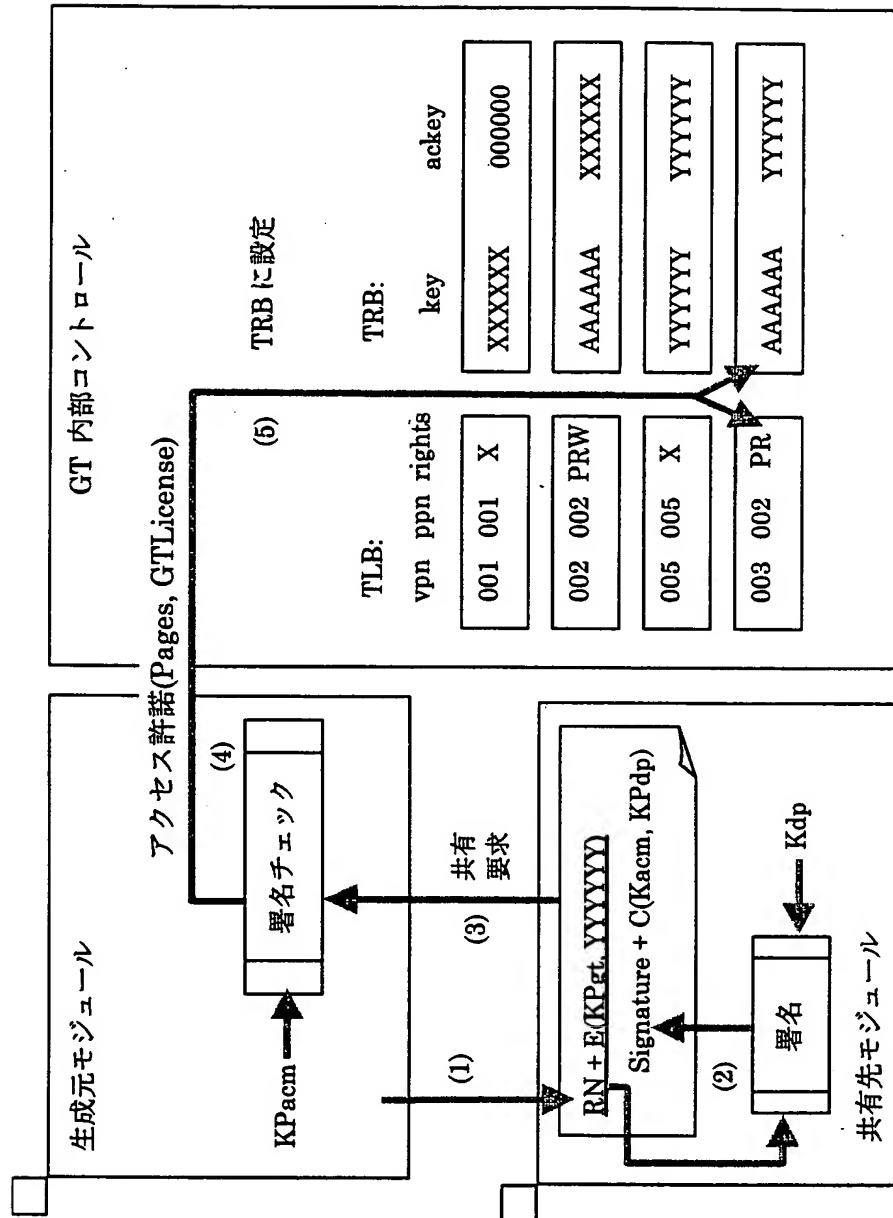


図 3 1



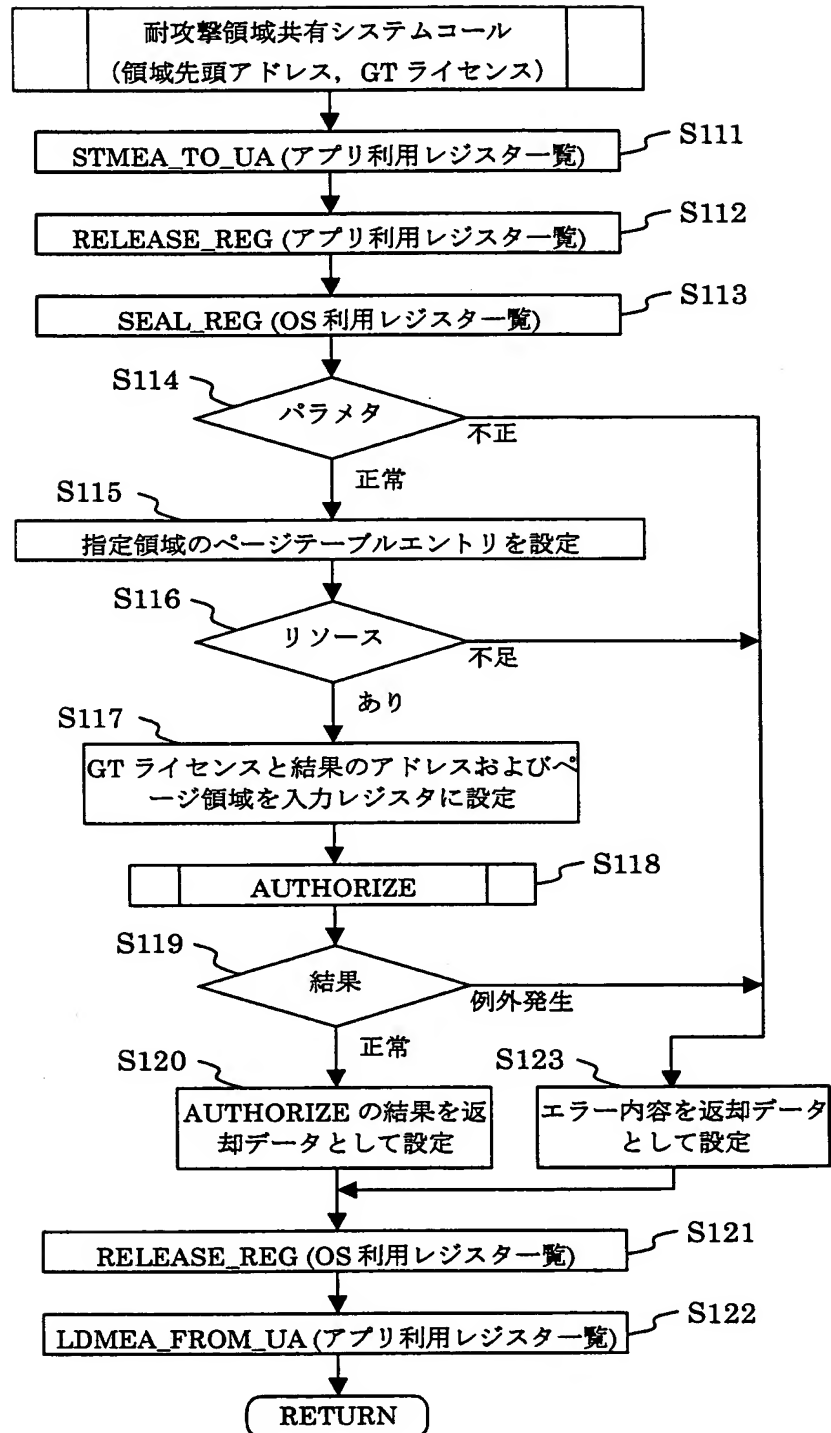


図 33

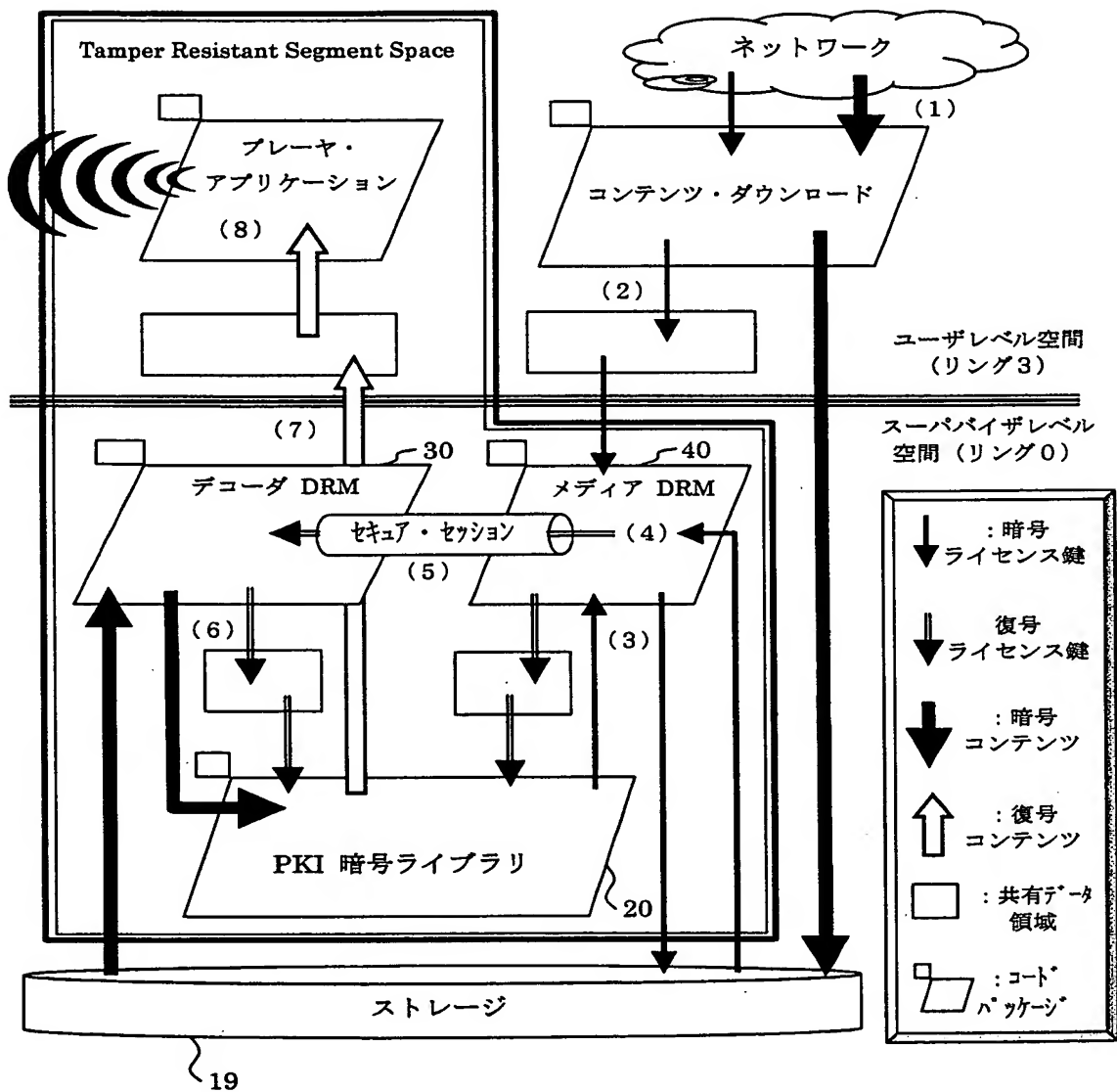


図 3 4

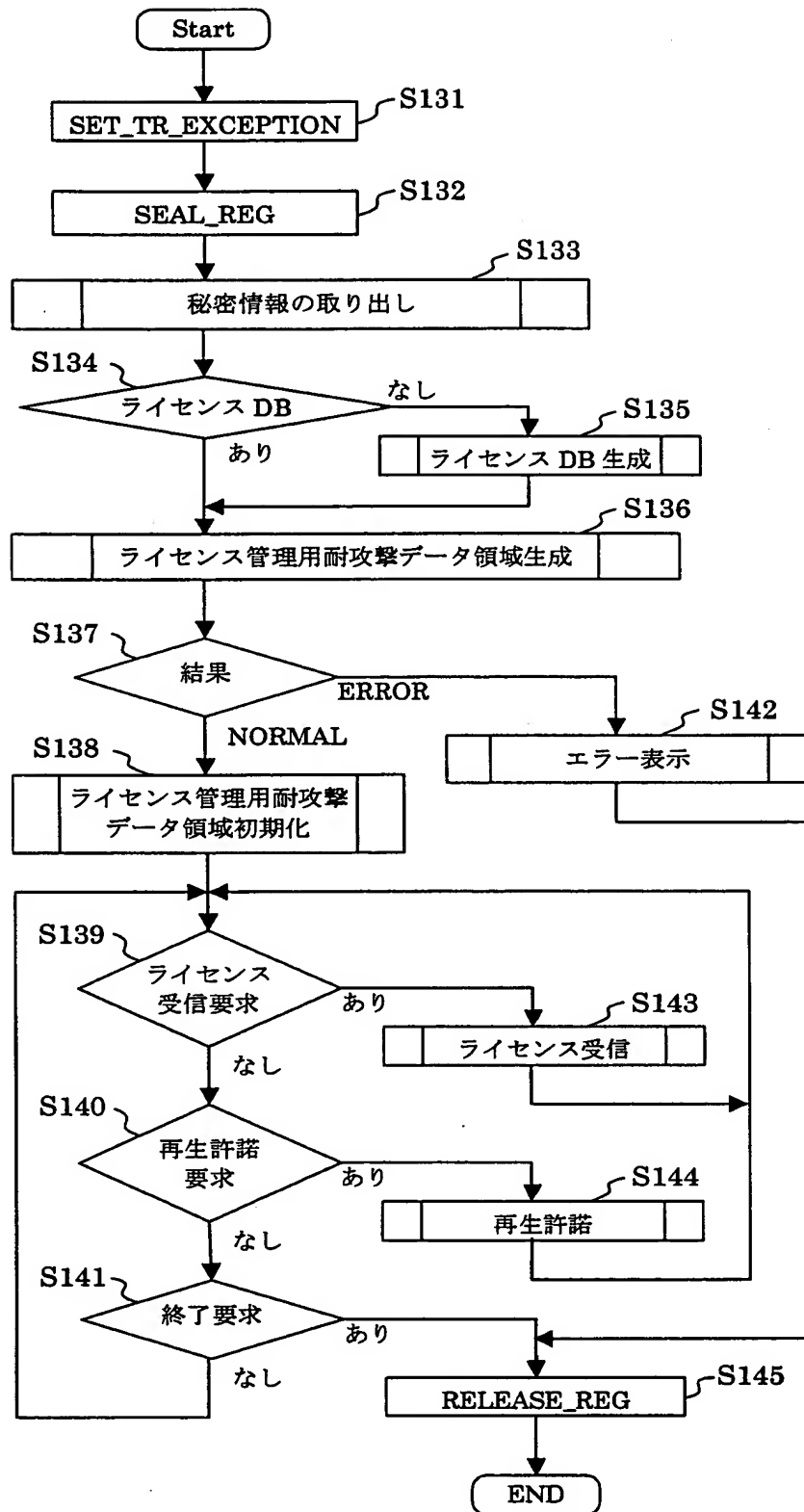


図 35



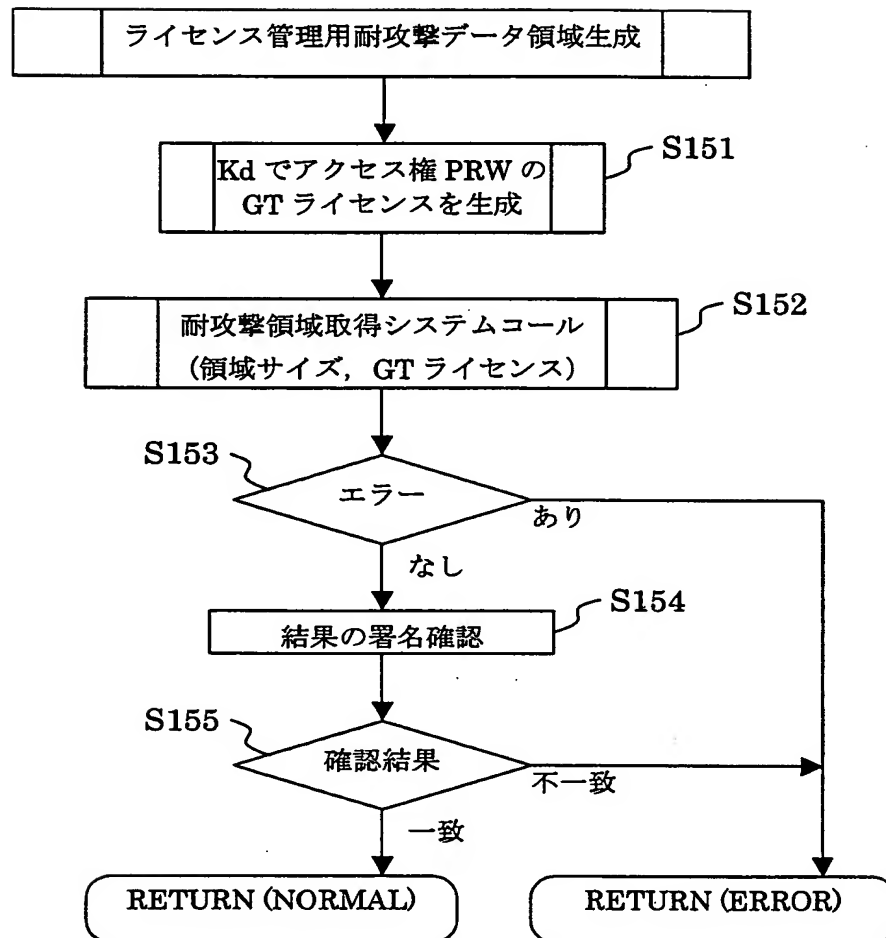


図 36

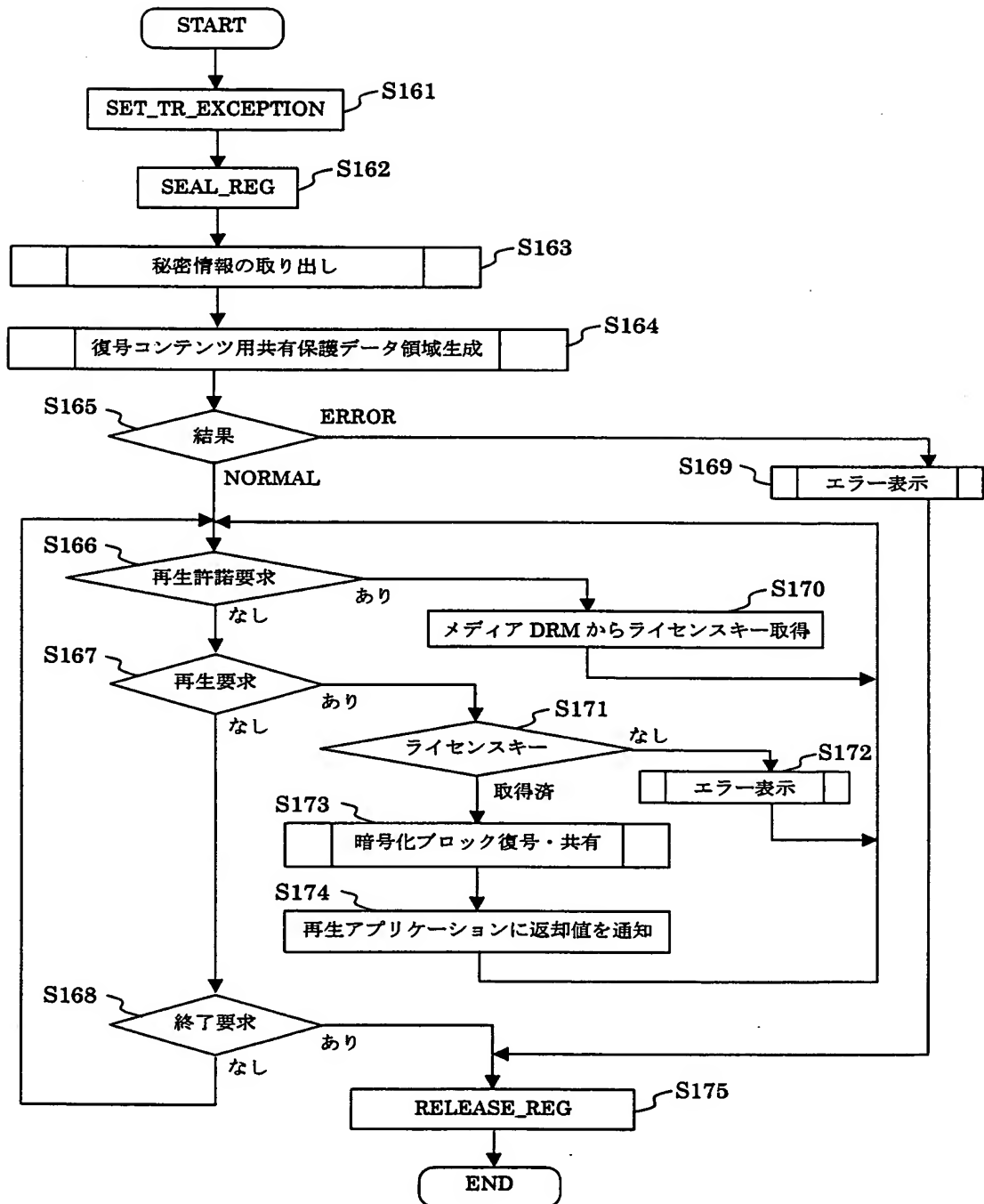


図 37

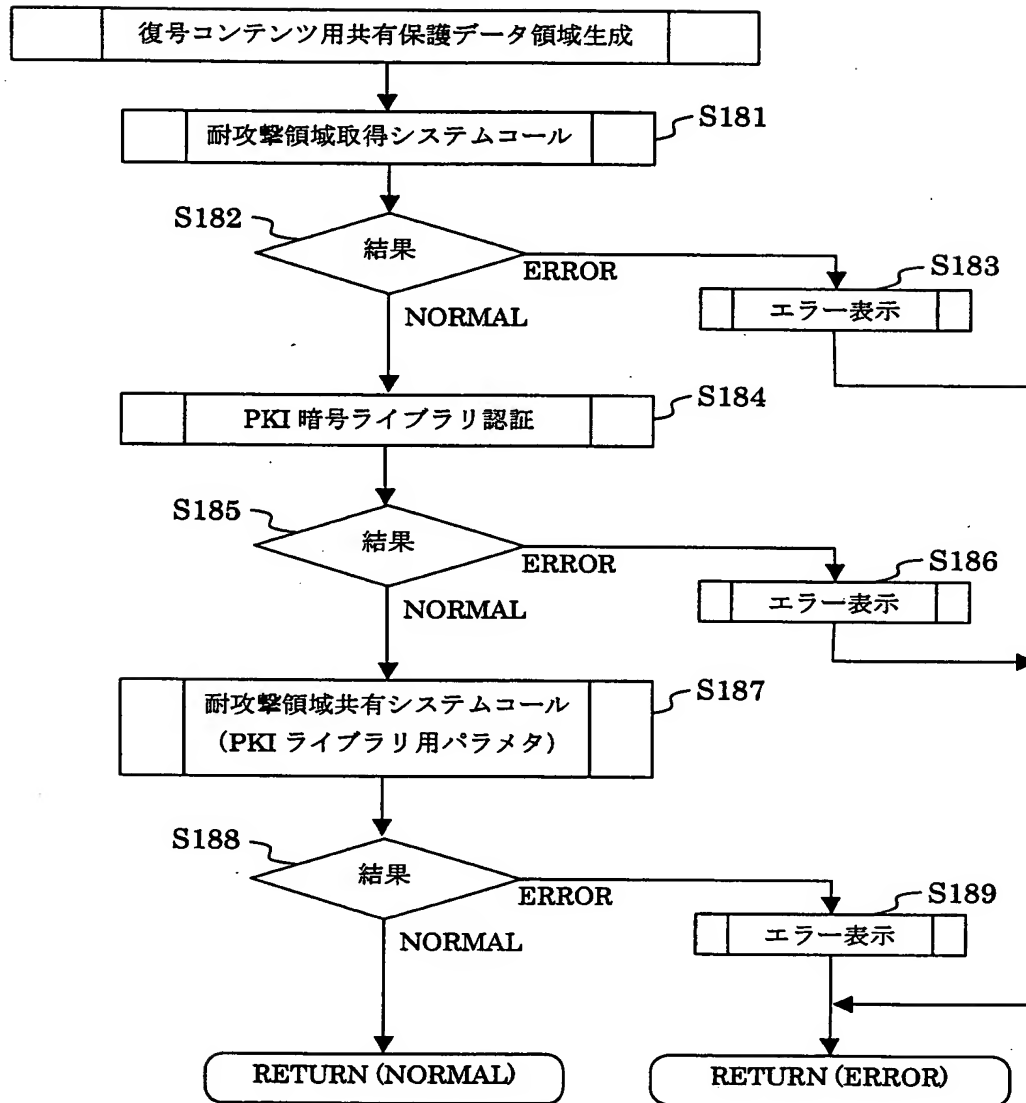


図 38

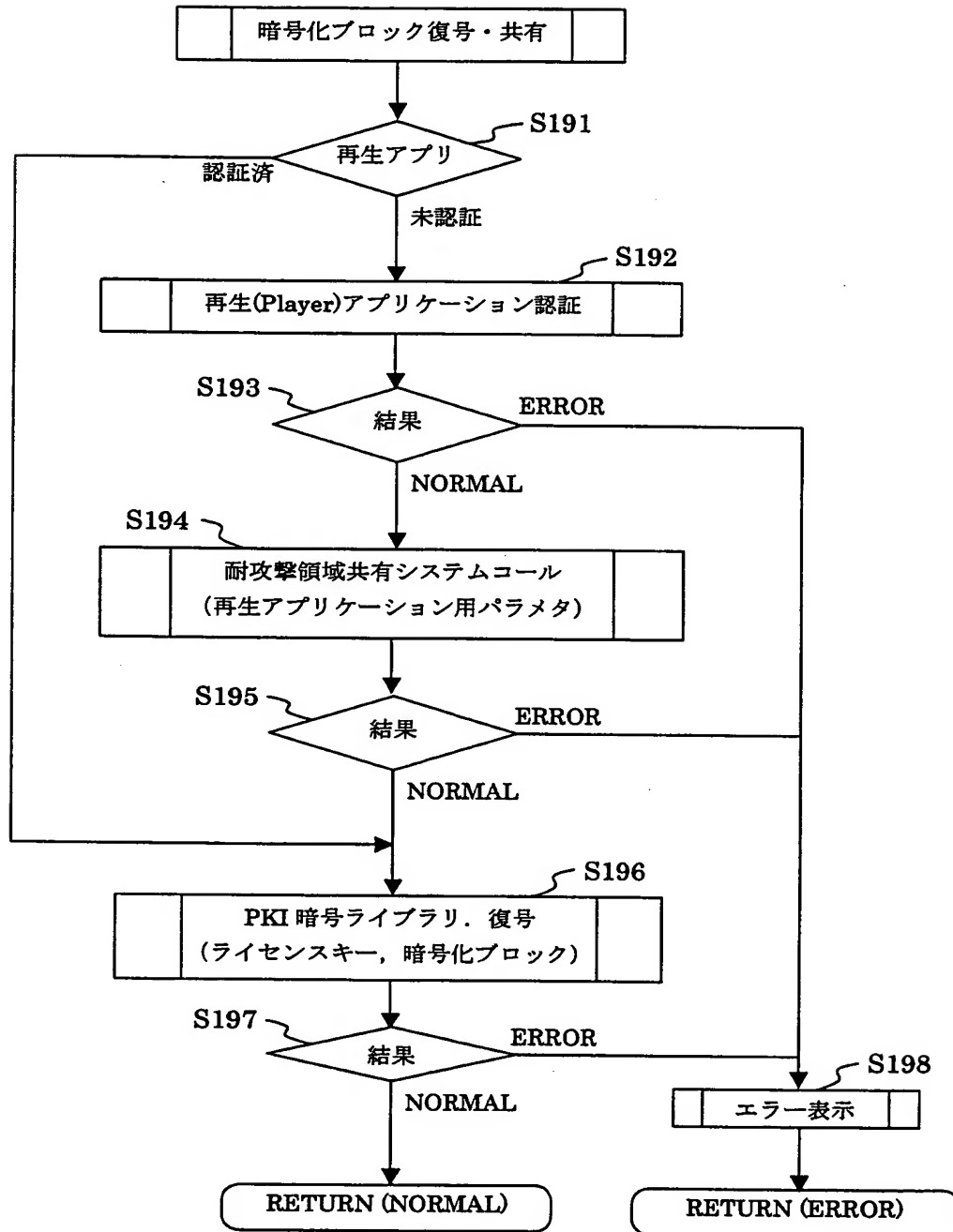


図 39

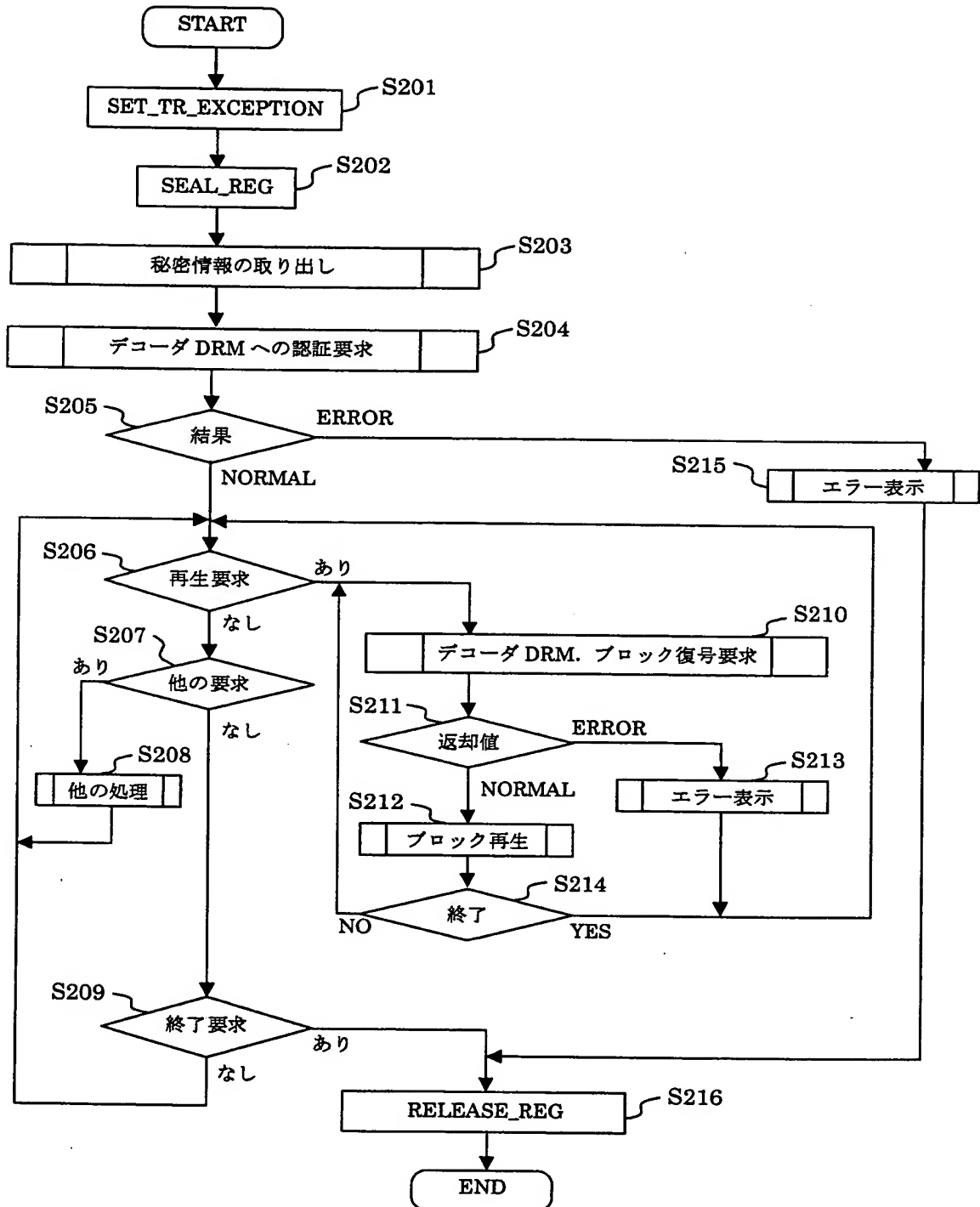


図 40

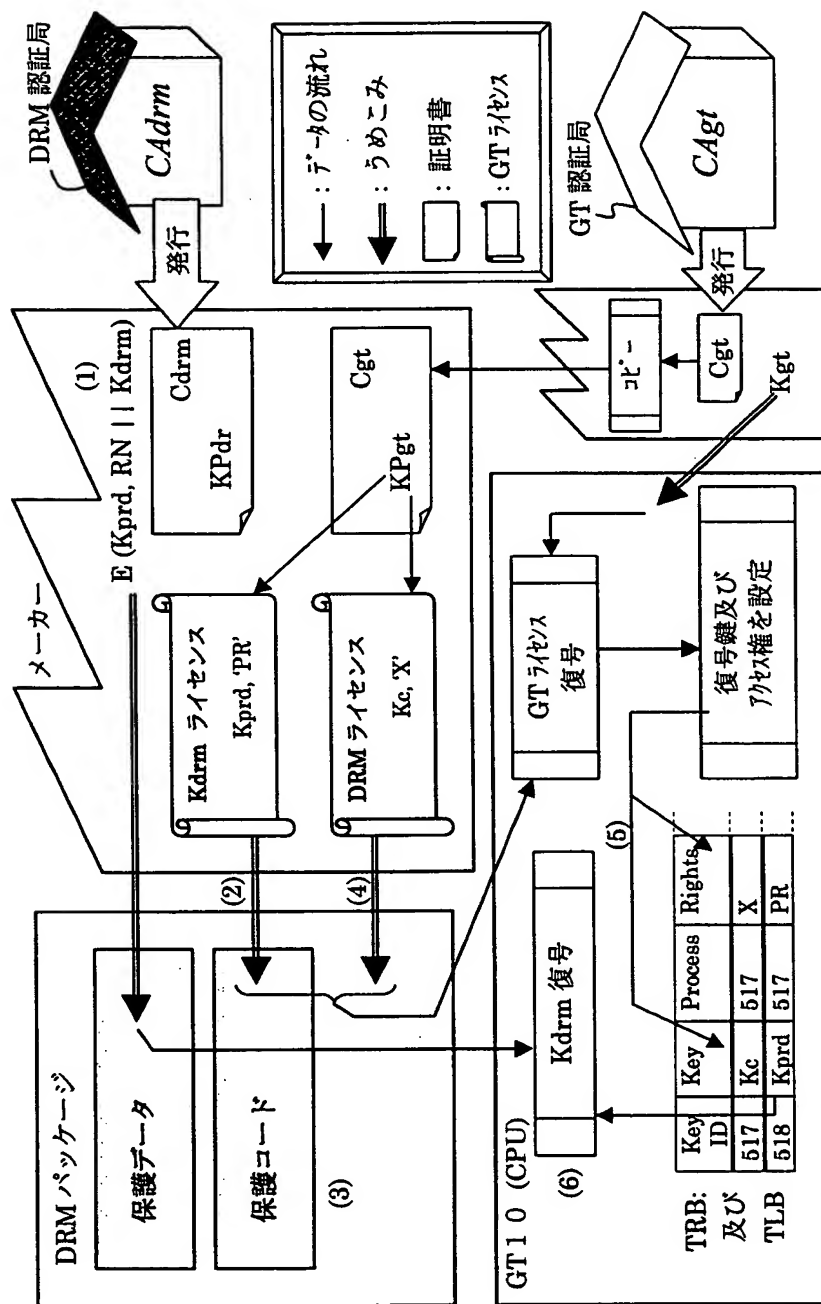


図 4 1

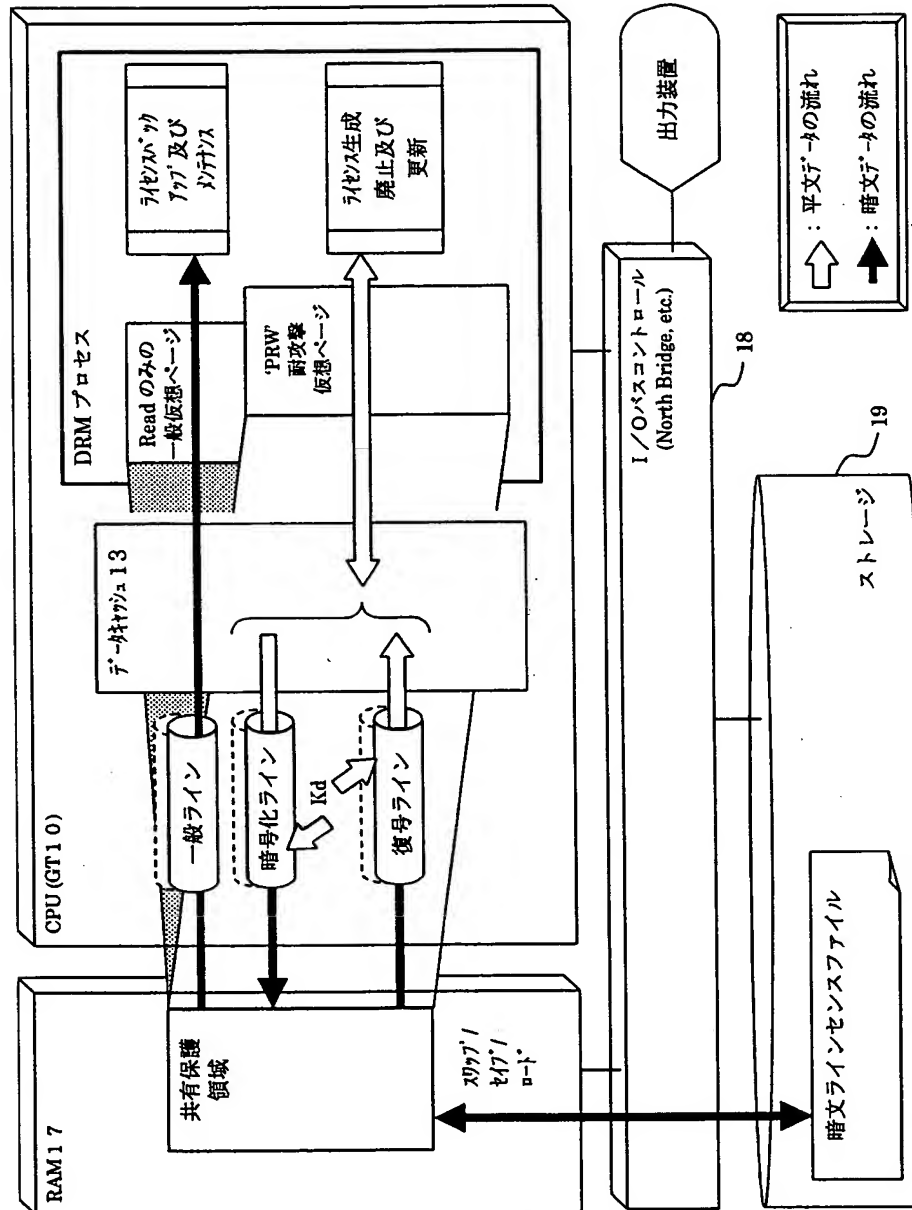


図 4 2

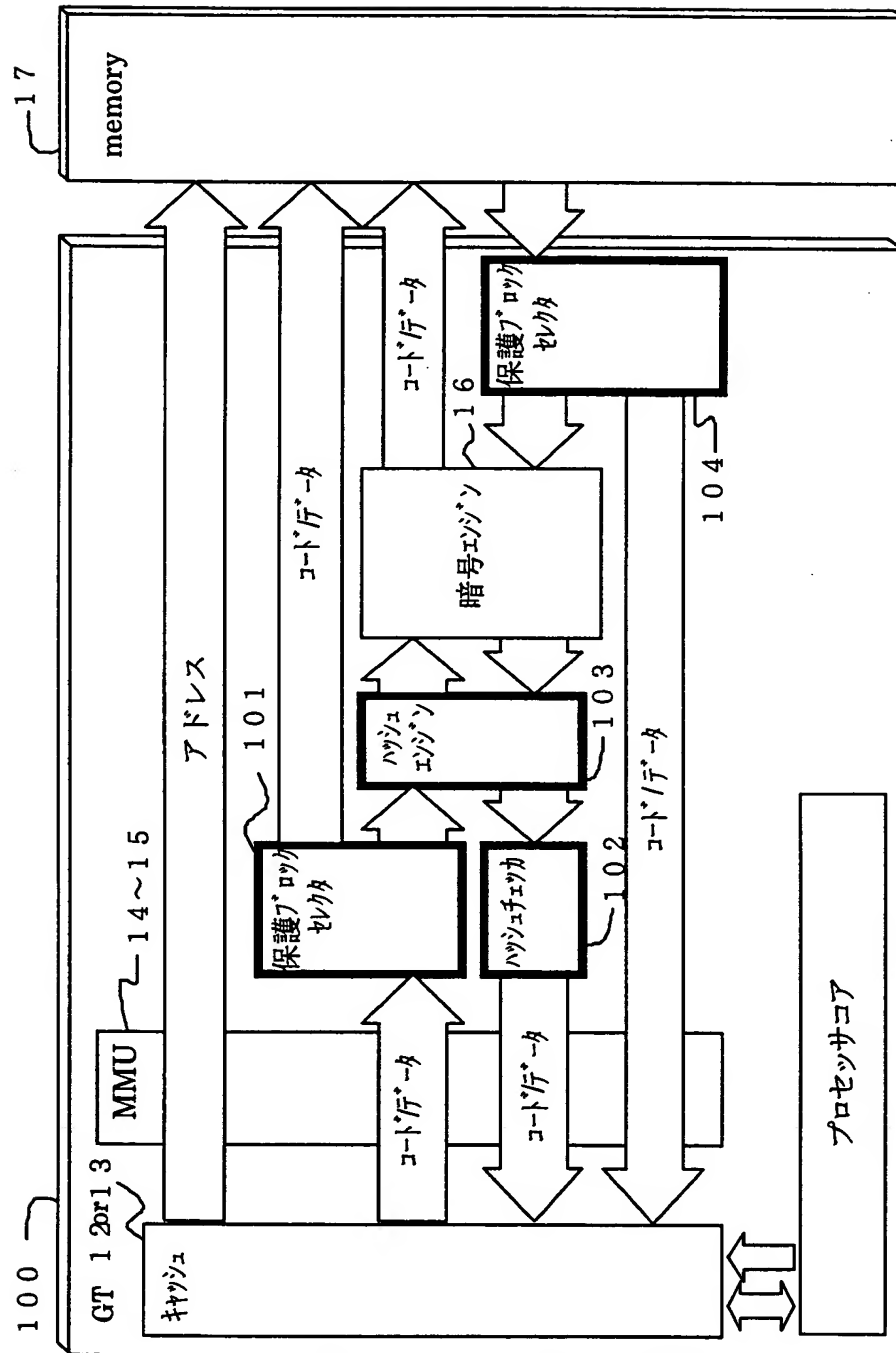
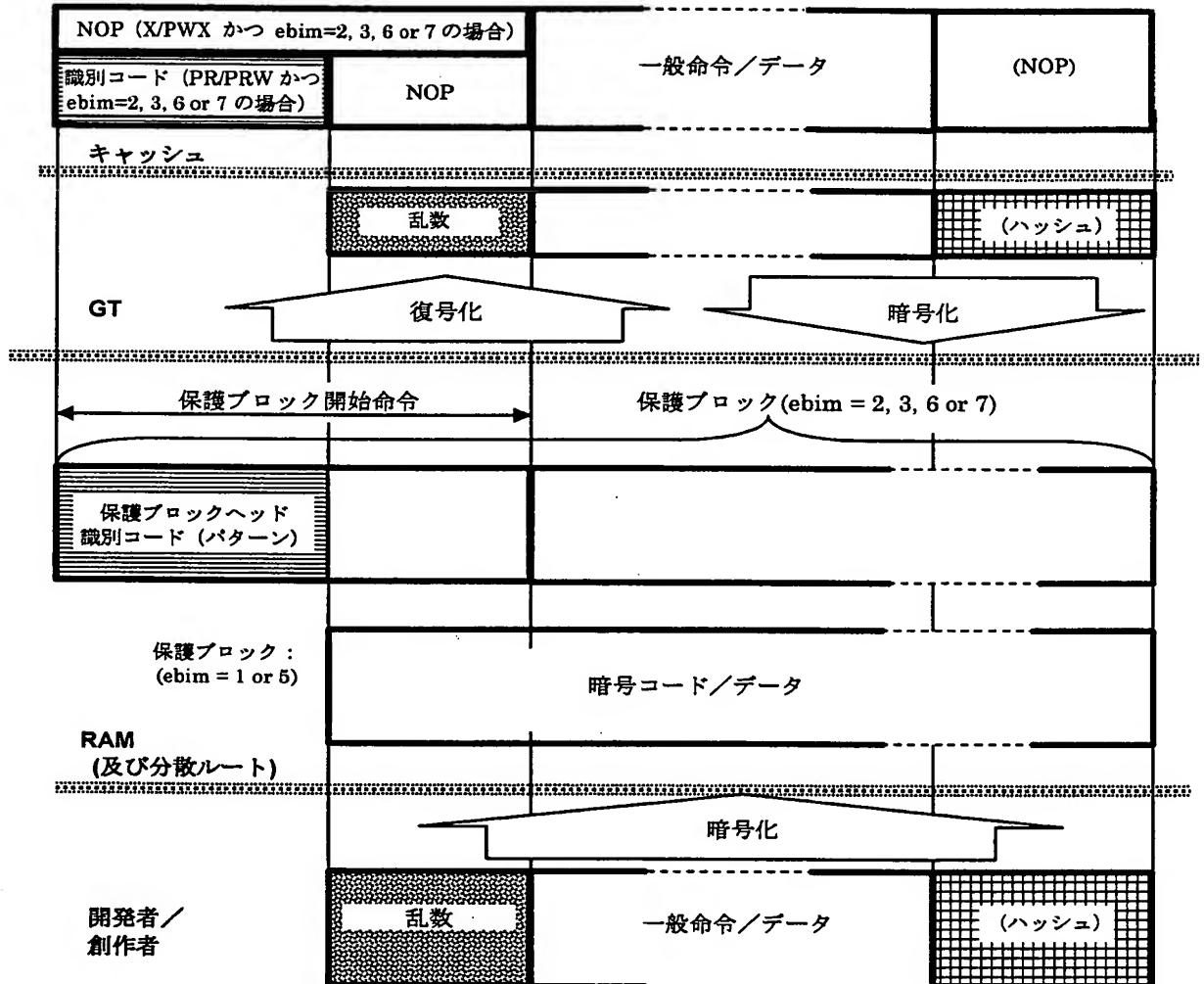


図 4 3





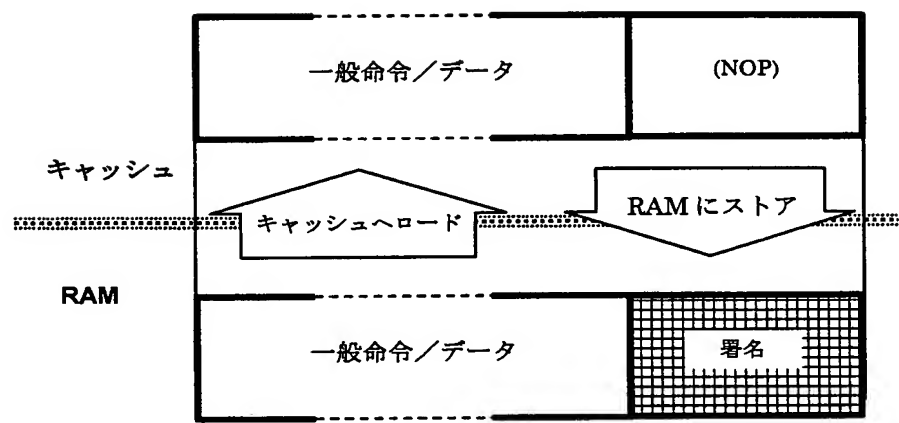


図 4 5

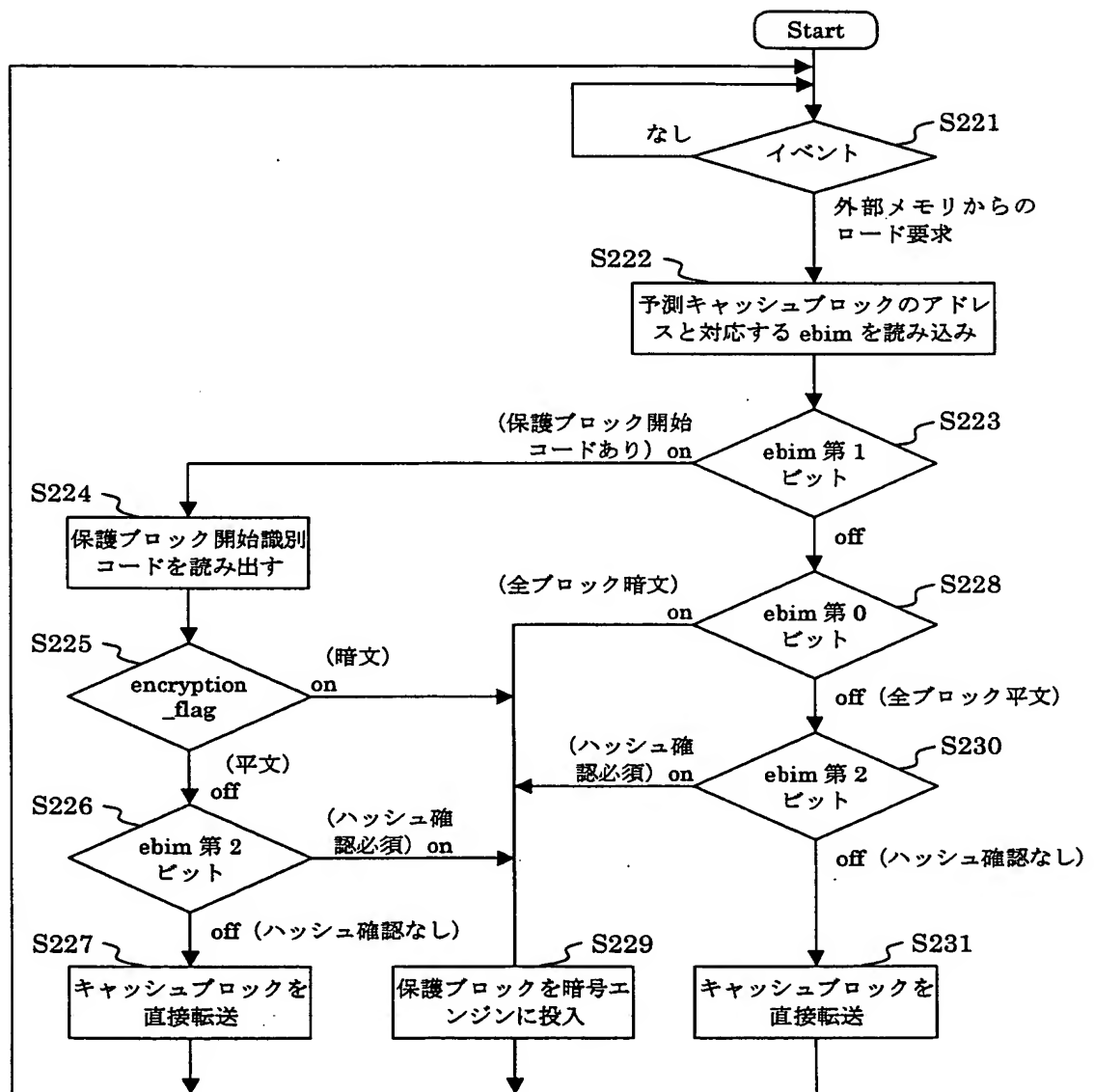


図 4 6

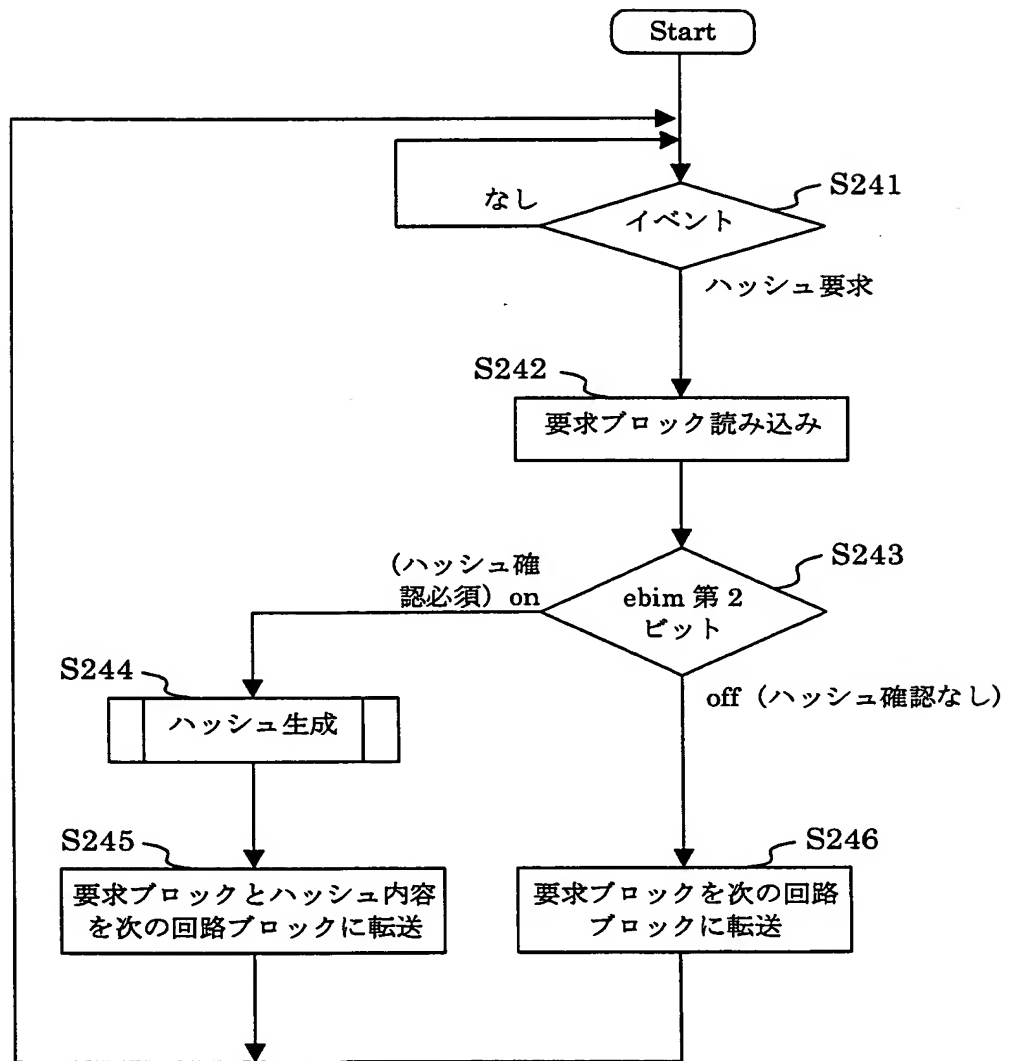


図 4 7

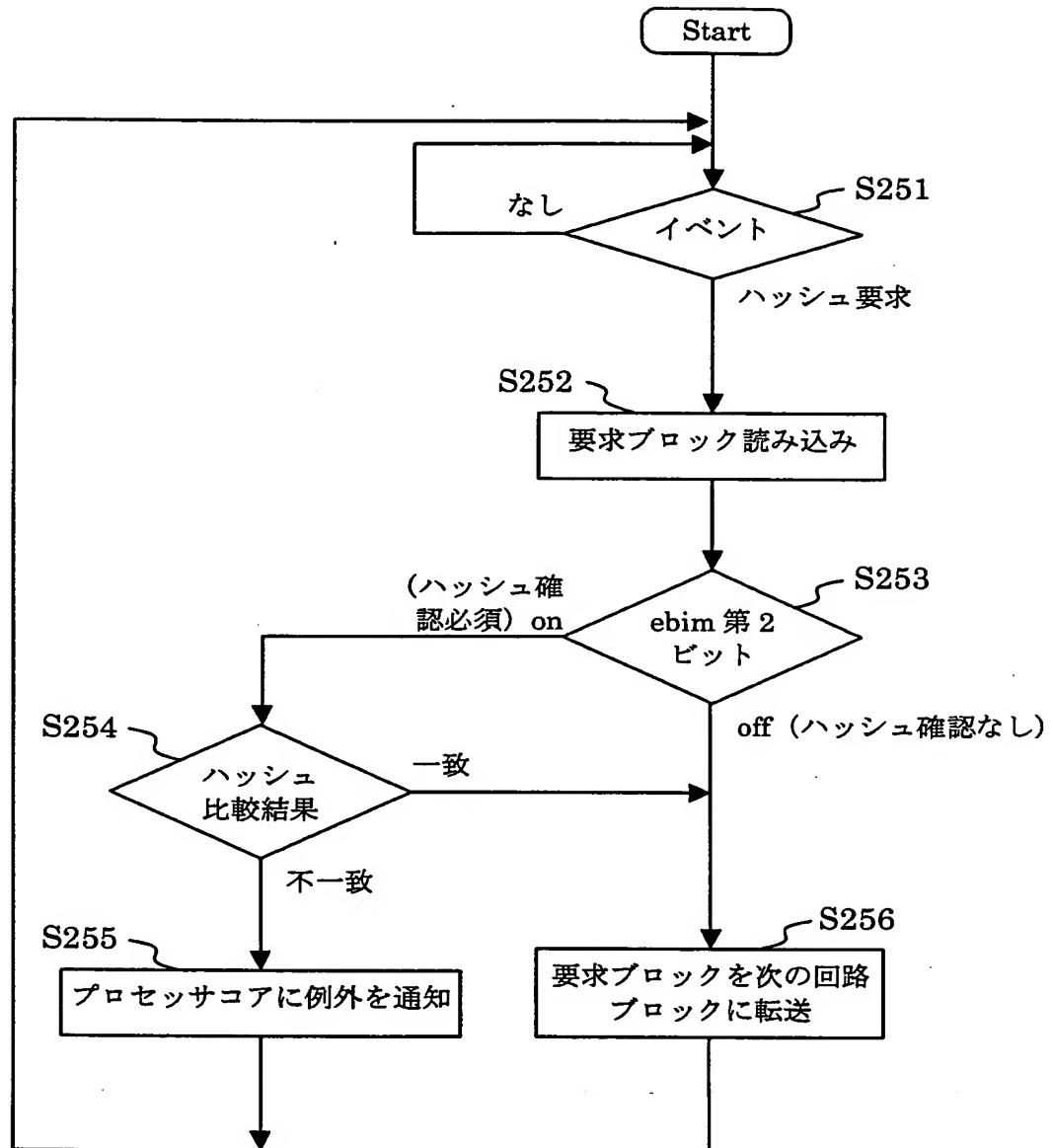


図 48

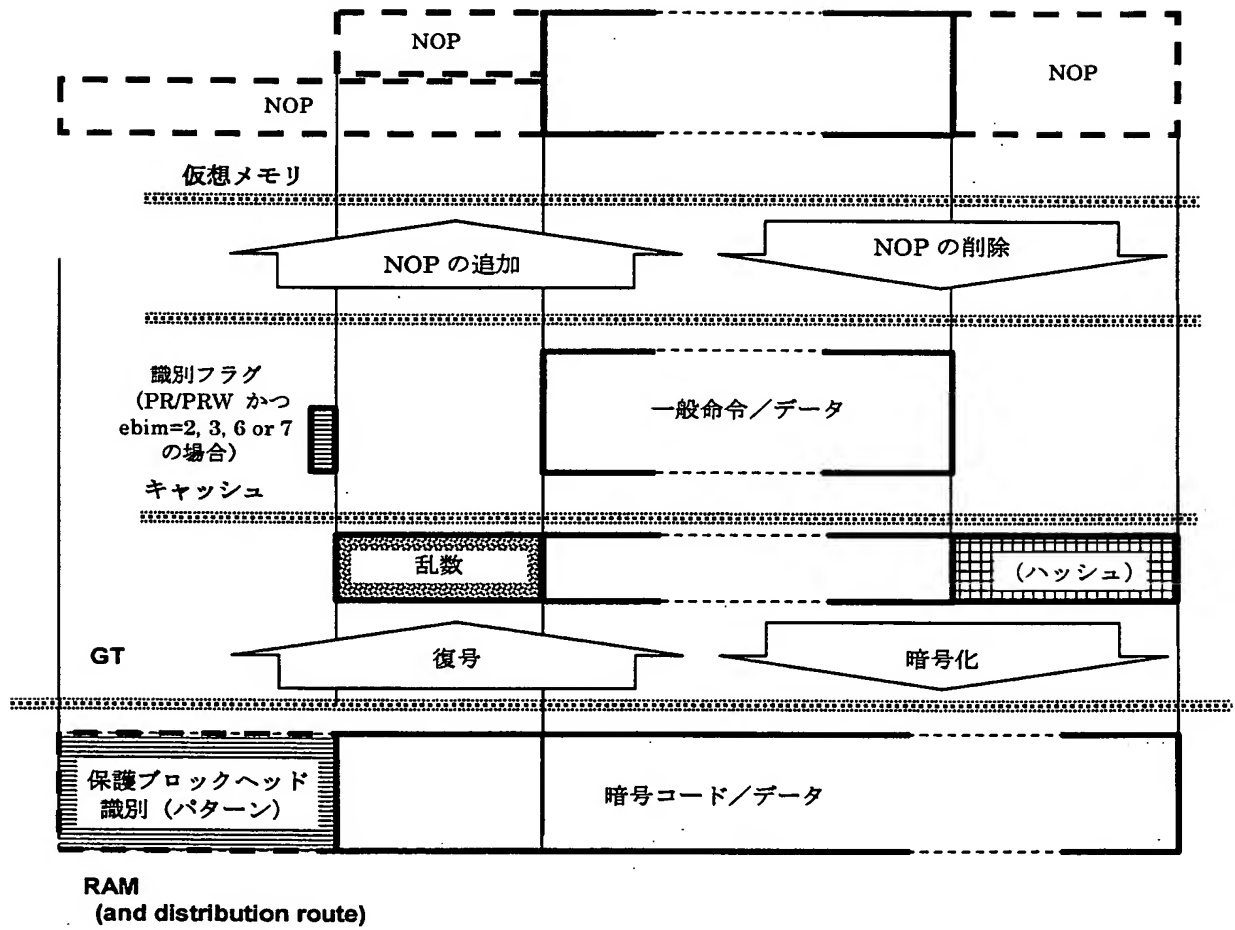


図 49

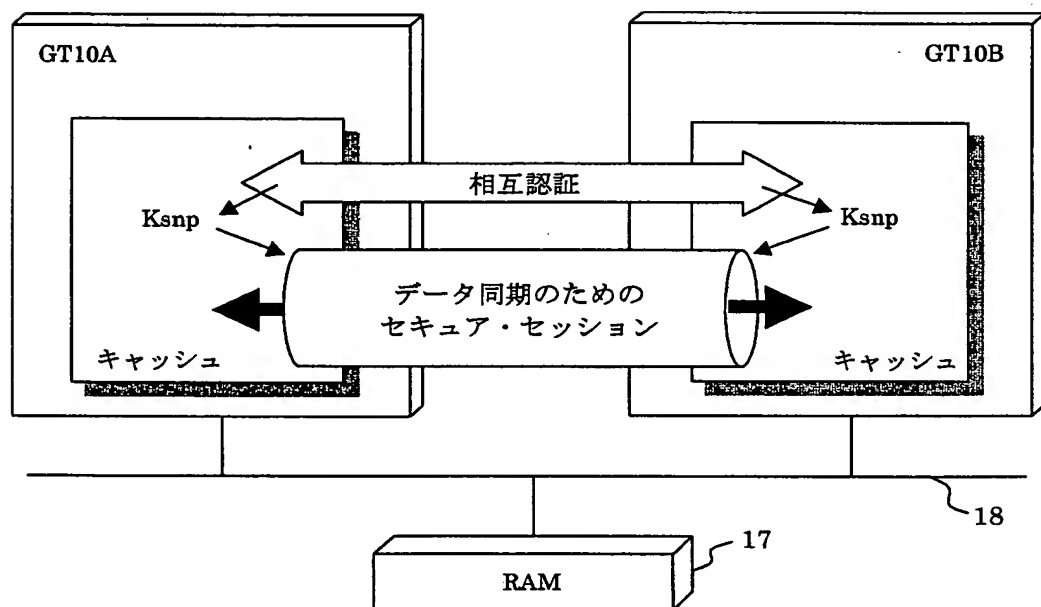


図 50

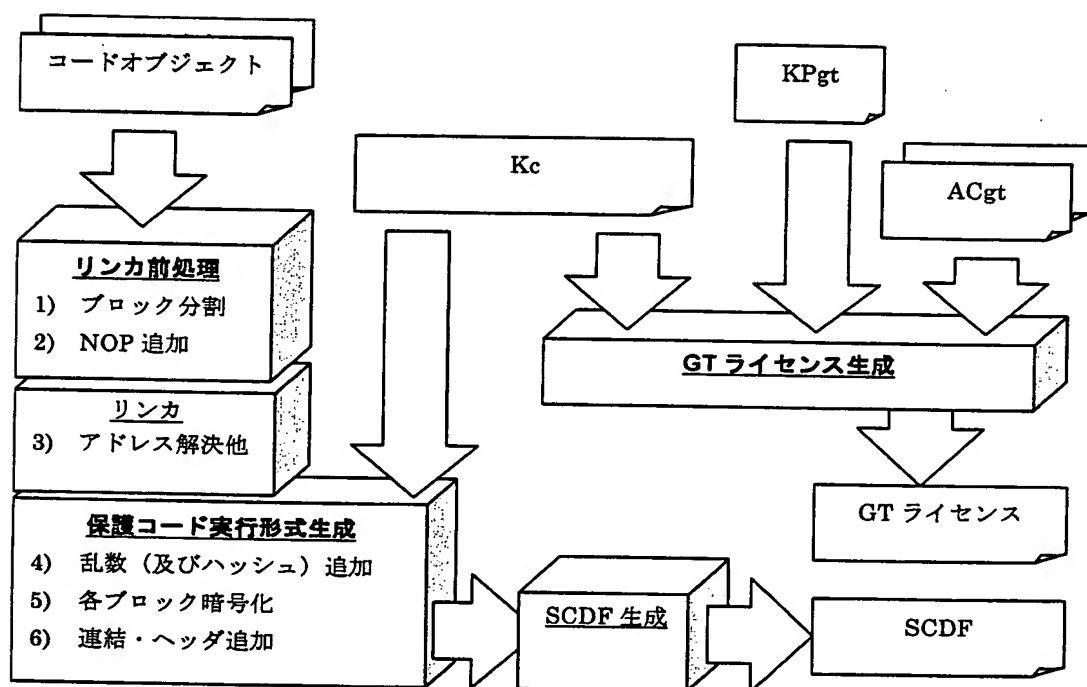


図 5 1



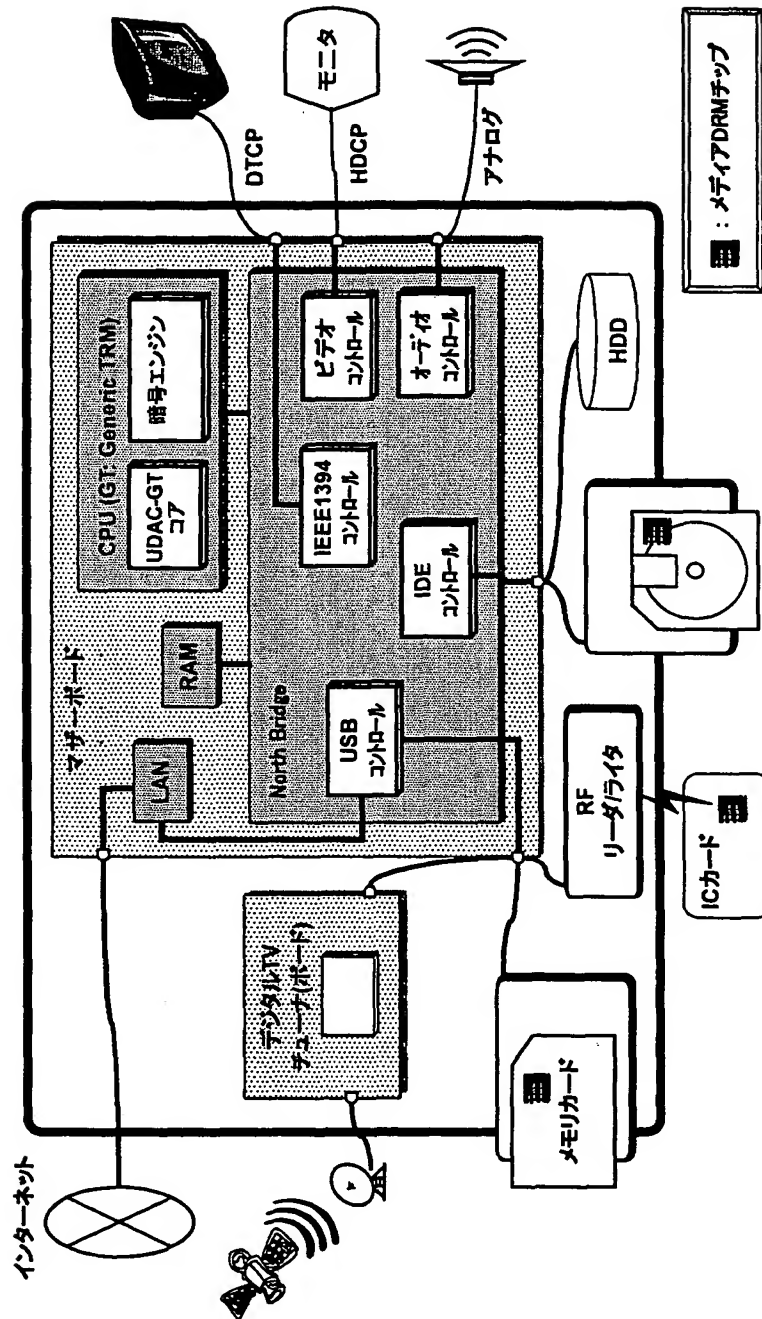


図 5 2